ulm university universität ulm

Universität Ulm | 89069 Ulm | Germany

Fakultät für Ingenieurwissenschaften, Informatik und Psychologie
Institut für Datenbanken und Informationssysteme

Entwicklung eines UI-Konzepts für das Datenmanagement in einem kollaborativen Task Management System

Bachelorarbeit an der Universität Ulm

Vorgelegt von:
Daniel Metzger
daniel.metzger@uni-ulm.de

Gutachter:
Prof. Dr. Manfred Reichert

Betreuer:
Nicolas Mundbrod

2018

Fassung 6. Mai 2018

Satz: PDF-LaTeX 2$_\varepsilon$

Kurzfassung

Der Bedarf an Wissen für Prozesse steigt weltweit immer stärker an. Um diesem Bedarf gerecht zu werden, werden Systeme benötigt, die Wissensarbeitern effektives Aufgabenmanagement bieten. Dabei muss dieses System die Charakteristiken wissensbasierter Prozesse berücksichtigen und diese in allen Phasen ihres Lebenszyklus unterstützen. Das proCollab Projekt der Universität Ulm widmet sich diesen Herausforderungen und bietet ein Lebenszyklusbasiertes Aufgabenmanagement. Dazu stellt proCollab den Wissensarbeitern Aufgabenlisten zur Verfügung mit denen sie ihre Aufgaben planen und koordinieren können. Ein umfangreiches Datenmanagement soll es Wissensarbeitern ermöglichen, Daten für Prozesse und Aufgaben auszutauschen und zu verwalten.

Dazu wird in dieser Arbeit ein User Interface-Konzept entwickelt, das die im Server implementierten Funktionen dem Benutzer zur Verfügung stellt. Dieses Konzept soll sich einheitlich und konsistent in den vorhanden proCollab Web-Client integrieren.

Inhaltsverzeichnis

1 Einleitung **3**

1.1 Problemstellung 3
1.2 Zielsetzung 4
1.3 Struktur der Arbeit 5

2 Grundlagen **7**

2.1 Wissensintensive Prozesse 7
2.2 proCollab 8
2.2.1 Lebenszyklus von wissensintensiven Prozessen 9
2.2.2 proCollab Schlüsselkomponenten 10
2.2.3 Workspace 13
2.2.4 Datenmanagement in proCollab 13
2.3 Usability 15
2.3.1 Grundsätze der Dialoggestaltung 16

3 Anforderungsanalyse **19**

3.1 Ist-Zustand 19
3.1.1 proCollab 19
3.1.2 Vergleichbare Systeme 22
3.2 Aufgabenanalyse 28
3.2.1 Datenmanagement von Data Spaces 28
3.2.2 Spezifizierte Input-, Output- und Hybrid-Datenelemente 39
3.2.3 Datenübersicht in Prozessinstanz und Prozess Template 45
3.2.4 Verwaltung strukturierter Daten Templates 49
3.3 Entwicklungskontext 53
3.3.1 Hardware 54
3.3.2 Software 54

4 Konzept **57**

4.1 Konzeptuelles User-Interface-Modell 57
4.1.1 Darstellungsregeln aus dem vorhandenen Client 57
4.1.2 Dialogstruktur 60
4.2 User-Interface Mockups 61
4.2.1 Grundsätzliche Darstellung von Datenelementen 61

4.2.2 Datenmanagement von Data Spaces 63
4.2.3 Input-, Output- und Hybrid-Datenelemente 67
4.2.4 Datenübersicht in Prozessinstanz und Prozess Template 69
4.2.5 Verwaltung strukturierter Daten Templates 69

5 Realisierung **73**
5.1 Verwendete Technologien 73
5.1.1 Angular 4 73
5.1.2 TypeScript 74
5.1.3 Sass 75
5.2 Ausgangssituation 75
5.3 Implementierung 77
5.3.1 Datenmanagement von Data Spaces 78
5.3.2 Modaler Dialog für Input-, Output- und Hybrid-Datenelemente . . . 83

6 Fazit **87**
6.1 Zusammenfassung 87
6.2 Ausblick 88

1 Einleitung

In entwickelten Ländern findet zunehmend ein Wandel von einer industriellen zu einer wissensbasierten Gesellschaft statt [MR14]. Schon bei kleineren Produkten wird ein komplexes Wissen über Entwicklung, Produktion, Marketing, Vertrieb und Logistik benötigt. Gleichzeitig nimmt durch die voranschreitende Globalisierung, der Konkurrenzdruck immer weiter zu. Um trozdem weiter bestehen zu können, sind Unternehmen dadurch gezwungen, ihre Produkte in immer kürzeren Abständen weiterzuentwickeln und ihre Produktionsverfahren zu verbessern. Diese Verbesserungen können allerdings nicht mit den Mitteln der verbesserten industriellen Produktion erreicht werden, sondern setzen den Einsatz von Wissen voraus [Hub05].

1.1 Problemstellung

Wissensarbeiter sehen sich immer komplexer werdenden wissensintensiven Prozessen gegenüber. Da diese aufgrund ihrer Komplexität und Dynamik nur schwer plan- und durchführbar sind, ist es notwendig auf Systeme zurückzugreifen, die Wissensarbeiter bei wissensintensiven Prozessen wirkungsvoll unterstützen können. Um ihre Ziele zu erreichen greifen Wissensarbeiter jedoch häufig auf einfache Aufgabenlisten, zum Beispiel in Form von Check- oder To-Do-Listen, zurück und nutzen zur Koordination bewährte Kommunikationsmittel wie zum Beispiel E-Mail [MBR15]. Diese Mittel zum Aufgabenmanagement sind allerdings fehleranfällig und wenig effizient und somit den wachsenden Ansprüchen von wissensintensiven Prozessen nicht mehr gewachsen.

Wissensintensive Prozesse lassen sich als unvorhersehbar, zielorientiert, entstehend und wissenerzeugend charakterisieren, wodurch das Planen und Durchführen von wissensintensiven Prozessen deutlich erschwert wird [MKR12]. Werden diese Eigenschaften von einem Informationssystem mit Check- und To-Do-Listen Unterstützung nicht vollständig einbezogen, oder existieren diese Listen sogar nur in Papierform, kann

dies Wissensarbeiter bei der Erreichung ihrer Ziele bedeutend beeinträchtigen. Des Weiteren ist es hilfreich, wenn Wissensarbeiter auf bereits bestehende Aufgabenlisten zurück greifen können, für den Fall, dass erneut ähnliche wissensintensive Prozesse auftreten. Das Zurückgreifen, oder gar Kombinieren von Aufgabenlisten aus vorangegangen Prozessen kann Wissensarbeitern bei neuen wissensintensiven Prozessen helfen.

Um Wissensarbeiter bei wissensintensiven Prozessen effizient in der Planung und Koordination unterstützen zu können, wurde das Projekt proCollab am Institut für Datenbanken und Informationssysteme der Universität Ulm im Jahr 2012 gestartet. proCollab bietet ein lebenszyklusbasiertes Aufgabenmanagement für wissensintensive Prozesse, das auf Prozesse, Aufgabenbäumen (engl. Task Trees) und Aufgaben (engl. Tasks) (siehe Kapitel 2.2.2) aufbaut und es ermöglicht für diese Vorlagen zu erstellen. Die Vorlagen sollen dem Ziel dienen die Planung und Koorditation durch *best practices* und definierten Standards zu beschleunigen [MBR15]. Wissensarbeiter können dann aus diesen Vorlagen diejenige instanziieren, welche am besten zur Erreichung ihrer Ziele geeignet ist.

Das notwendige Wissen welches Wissensarbeiter für Prozesse und Aufgaben benötigen wird häufig in Dateien jeglicher Art, zum Beispiel in PDFs, Office-Dokumente oder Videoaufzeichnungen festgehalten. Dafür bietet proCollab ein umfangreiches Datenmanagement, das es Wissensarbeitern ermöglicht an Prozessen und Aufgaben die zur Ausführung benötigten Daten einzubinden, aber auch neue Daten, die während der Ausführung oder nach Abschluss eines Prozesses oder einer Aufgabe entstehen, daran anzuhängen.

1.2 Zielsetzung

Das Datenmanagement ist im proCollab-Server bereits implementiert. Ziel dieser Arbeit ist die Entwicklung und Implementierung eines User Interface-Konzepts (kurz UI-Konzepts) für das proCollab Datenmanagement. Genauer soll ermöglicht werden Datenelemente an die proCollab Systemkomponenten, Prozesse, Task Trees und Tasks anzuhängen und mit ihnen zu interagieren.

Um den entstehenden und unvorhersehbaren Charakter von wissensintensiven Prozessen gerecht zu werden, existieren in proCollab zwei Arten von Datenelementen. Dies sind zum Einen unstrukturierte Daten, wie Word- oder PDF-Dateien und zum Anderen strukturierte Daten, z.B. Adressen mit den Attributen Name, Straße, Postleitzahl und Ort.

Diese Datenelemente sollen nicht nur in Systemkomponenten abgelegt werden, sondern auch als Startvoraussetzung (Input), bzw. als Ergebnis (Output) eines Arbeitsschrittes, im Projekt festgelegt werden können.

Die serverseitige Implementierung, der für das Datenmanagement notwendigen Funktionen, ist vorhanden, die Konzepte müssen nun für den Client-Ansatz von proCollab entwickelt werden. Dazu wird ein Darstellungskonzept benötigt, dass das Hinzufügen, Darstellen und Interagieren mit strukturierten und unstrukturierten Daten ermöglicht. Bevor strukturierte Daten aber überhaupt verwendet werden können, müssen diese von einem Nutzer erstellt und verwaltet werden, dafür muss also ebenfalls ein UI-Konzept entwickelt werden. Weiter muss eine Lösung für die Darstellung von Input- und Output-Datenelementen gefunden werden.

Diese Arbeit stellt Konzepte vor, die diese Anforderungen effizient, konsistent und unter Berücksichtigung von gängigen Usability-Richtlinien zur Dialoggestaltung in das bestehende proCollab User Interface integriert und dieses sinnvoll erweitert.

1.3 Struktur der Arbeit

Im ersten Kapitel wurde das Problem beschrieben und das Ziel der Arbeit herausgearbeitet. Kapitel 2 liefert notwendige, für das Verständnis der Arbeit benötigte Grundlagen, erläutert die wichtigsten Komponenten von proCollab und liefert einen Überblick über Usability und welche Grundsätze in der Dialoggestaltung in dieser Arbeit angewendet werden. In Kapitel 3 wird schließlich der aktuelle Stand von proCollab vorgestellt und mit ähnlichen Systemen verglichen. Die Anforderungen an das UI-Konzept werden schließlich in der Kontextanalyse konkretisiert. Unter Berücksichtigung der im dritten Abschnitt ermittelten Informationen, wird in Kapitel 4 zuerst ein konzeptuelles User Interface-Modell entwickelt, welches grundlegende Gestaltungsregeln für das im nächsten Schritt zu gestaltende Konzept definiert. Kapitel 5 widmet sich der Implementierung des entwickelten Konzeptes in den proCollab Client. Im sechsten und letzten Kapitel dieser Arbeit findet eine Diskussion über die Ergebnisse, Limitationen und eventuelle nachfolgende Arbeiten statt.

2

Grundlagen

In Kapitel 2 werden die für das Verständnis dieser Arbeit notwendigen Grundlagen behandelt. Dabei wird in Kapitel 2.1 der Begriff *wissensintensive Prozesse* definiert und deren Charakteristiken erläutert. In Kapitel 2.2 wird der Aufbau von proCollab dargestellt, dabei wird auf den Lebenszyklus wissensintensiver Prozesse und die wesentlichen Schlüsselkomponenten, sowie dem Data Repository eingegangen. Abschließend wird in Kapitel 2.3 *Usability* und die *Grundsätze der Dialoggestaltung* definiert und erläutert wie sie in diese Arbeit einfließen werden.

2.1 Wissensintensive Prozesse

Im Gegensatz zu standartisierten Prozessen, handelt es sich bei wissensintensiven Prozessen wegen ihrer Dynamik um eine andere Form von Prozessen. Da verschiedene Definitionen von wissensintensiven Prozessen existieren [DMR14], bezieht sich diese Arbeit auf folgende in [Vac11] zugrunde liegende Formulierung:

> *"Knowledge-intensive processes are processes whose conduct and execution are heavily dependent on knowledge workers performing various interconnected knowledge-intensive decision making tasks. KiPs are genuinely knowledge, information and data-centric and require substantial flexibility at design- and run-time."*

Des Weiteren besitzen wissensintensive Prozesse folgende vier wesentliche Charakteristiken [MKR12]:

Unvorhersehbarkeit

Wissensarbeit unterliegt aufgrund ihrer Komplexität einer großen Anzahl sich dynamisch ändernder Einflussfaktoren. Diese lassen sich von Wissensarbeitern kaum bis in das

letzte Detail vorhersehen und planen. Durch diese Dynamik ist es notwendig, sowohl die Planung als auch die Ausführung wiederholt neu zu bewerten und hinsichtlich den neuen Gegebenheiten anzupassen.

Zielorientierung

Wissensarbeiter benötigen ein gemeinsames Ziel, das das Endergebnis ihrer Arbeit definiert. Für sehr komplexe wissensintensive Prozesse können leichter zu erreichende Zwischenziele behilflich sein. Während das gemeinsame Ziel nicht geändert werden soll, kann ein Zwischenziel verändert oder sogar wieder entfernt werden.

Prozessentstehung

Aktivitäten unterliegen ständigen Änderungen durch Wissensarbeiter, die auf Grund von Einflussfaktoren Anpassungen vornehmen müssen, um ihre Zwischenziele erreichen zu können. Aus dieser Dynamik ergibt sich, dass durch Wissensarbeiter nur die unmittelbar anstehenden Aktivitäten genau geplant werden können, während weiter entfernte Aktivitäten noch nicht oder nur grobgranular bedacht werden.

Wissensarbeiter gehen im Umgang mit wissensintensiven Prozessen nach dem Plan-Do-Study-Act Prinzip vor [MKR12, MR14]. Dabei durchlaufen sie die vier Stufen, Arbeitsplanung, Arbeitsausführung, Evaluation der Arbeitsergebnisse und zuletzt Optimierung der Arbeitsplanung basierend auf den neu gewonnenen Erkenntnissen [MR17].

Wachsende Wissensbasis

Wissensarbeiter nutzen ihr Wissen und benötigen Informationen um ihre Ziele erreichen zu können. Neu erlangte Informationen müssen bewahrt werden und gegebenenfalls als Basis zur Erreichung neuer Ziele zur Verfügung stehen. Dafür ist eine stetig wachsende Wissensbasis und eine effektive Verwaltung notwendig.

In wissensintensiven Prozessen kann es notwendig sein auf Informationen oder Dokumente zurückzugreifen, die aus vorangegangenen Aufgaben zur Verfügung stehen.

2.2 proCollab

Um Wissensarbeiter bei wissensintensiven Prozessen in der Planung und Koordination unterstützen zu können, wurde das Projekt proCollab am Institut für Datenbanken und

Informationssysteme an der Universität Ulm von Prof. Dr. Manfred Reichert und Nicolas Mundbrod im Jahr 2012 gegründet. Dabei berücksichtigt proCollab die in Kapitel 2.1 beschriebenen dynamischen und unvorhersehbaren Eigenschaften wissensintensiver Prozesse und soll diese in allen Phasen auf Basis eines Lebenszyklusbasierten Ansatzes unterstützen.

2.2.1 Lebenszyklus von wissensintensiven Prozessen

Der Lebenszyklus von wissensintensiven Prozessen besteht aus vier unterschiedlichen Phasen (siehe Abbildung 2.1), die von Wissensarbeitern durchlaufen werden. Im Folgenden werden die Phasen des Lebenszyklus, wie sie in [MKR12] vorgestellt werden, beschrieben.

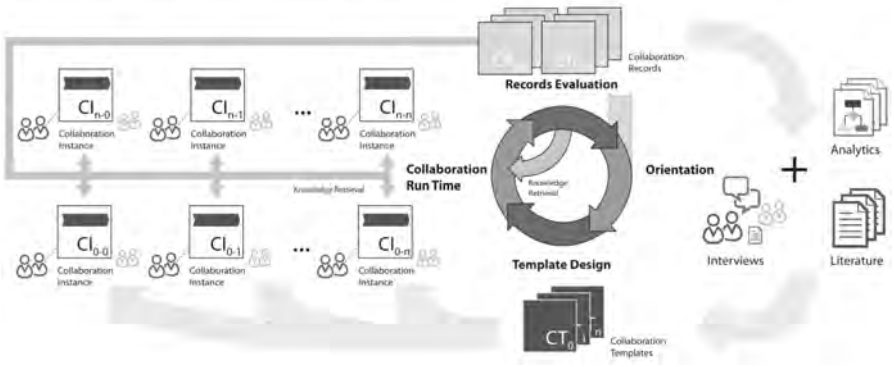


Abbildung 2.1: Der Lebenszyklus wissensintensiver Prozesse [MKR12]

Orientation Phase

Wissensarbeiter sammeln im ersten Schritt Informationen. Dazu werden die Ergebnisse ähnlicher abgeschlossener Arbeiten analysiert, Wissensarbeiter befragt und auf bestehende Literatur zurückgegriffen. Es soll festgehalten werden, welche Aufgaben die einzelnen Wissensarbeiter durchführen, wie sie miteinander kommunizieren und wie die gesammelten Informationen dokumentiert werden sollen.

Template Design Phase

Aus dem Informationsfluss wird nun ein Template erstellt, das anschließend für bestimmte wissensintensive Prozesse herangezogen werden kann. Das Template stellt Informationen, Kommunikations- und Koordinationsmöglichkeiten bereit, die Wissensarbeiter wahrscheinlich für ihre Arbeit benötigen. Dabei schreibt ein Template weder die Aktivitäten, noch deren Reihenfolge vor. Es soll so entworfen sein, dass es Wissensarbeiter bei ihrer Arbeit so gut es geht unterstützt, jedoch ohne sie dabei einzuschränken. Darüber hinaus soll es einen hohen Grad an Anpassbarkeit besitzen.

Collaboration Run Time Phase

Wissensarbeiter können nun ein Template, das am besten zu dem Kontext ihres angehenden wissensintensiven Prozesses passt, instanziieren und so eine Prozessinstanz erstellen. Es ist dabei sehr wichtig, dass sich die Prozessinstanz durch Wissensarbeiter leicht an ihre Bedürfnisse oder sich ändernde Bedingungen anpassen lässt.

Records Evaluation Phase

Records sind eine wichtige Informationsquelle für Wissensarbeiter. Die Informationen, die aus abgeschlossenen wissensintensiven Prozessen gewonnen werden, können für zukünftige Arbeiten herangezogen werden und als Basis für neue Templates dienen, oder aber bestehende verbessern.

2.2.2 proCollab Schlüsselkomponenten

Um Wissensarbeiter optimal bei ihrer Arbeit unterstützen zu können, besteht proCollab aus den drei Schlüsselkomponenten (siehe Abbildung 2.2) *Prozesse* (engl. *Processes*), *Task Trees* (engl. für *Aufgabenbäume*) und *Tasks* (engl. für *Aufgaben*) [MBR15]. Zusätzlich gibt es für jede Schlüsselkomponente *Vorlagen* (engl. *Templates*) und *Instanzen* (engl. *Instances*).

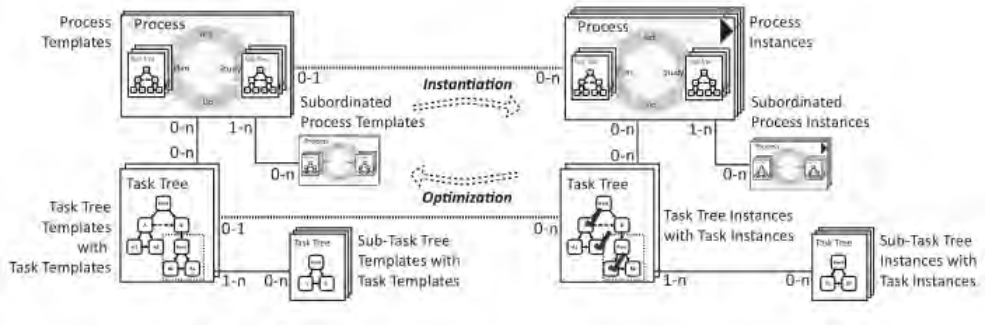


Abbildung 2.2: proCollab Übersicht [MR17]

Prozess

Ein Prozess kann ein Projekt, zum Beispiel die Entwicklung einer Webseite, aber auch ein Fall (engl. Case), zum Beispiel eine medizinische Behandlung, sein. Alle Prozesse haben einige gemeinsame Elemente: So liegt jedem Prozess ein definiertes Ziel zu Grunde, das Wissensarbeiter erreichen möchten, Rahmenbedingungen wie eine Deadline oder die verfügbaren Ressoucen. Weitere Gemeinsamkeiten sind Datenelemente und die Rollenverteilung mit den dazugehörigen Rechten innerhalb des Projekts oder des Cases. Um Wissensarbeiter bei dem Erreichen ihrer Ziele zu unterstützen verlinkt jeder Prozess, als wichtigstes Element, zu einem oder mehreren Task Trees.

Task Tree

Mit Hilfe von Task Trees lassen sich generisch jegliche Formen von Aufgabenlisten abbilden. In proCollab können unter anderem prospektive To-do Listen eingesetzt werden, um Aufgaben im Rahmen von wissensintensiven Prozessen zu planen und zu organisieren. Retrospektiv helfen Checklisten den Wissensarbeitern dabei, die Qualität ihre Arbeit sicherzustellen. Dabei müssen sich Wissensarbeiter nicht zwingend an die vorgegebene Reihenfolge der Listenelemente halten, sondern können diese entsprechend der Erforderlichkeiten des wissensintensiven Prozesses wählen. Ein Task Tree ist eine generische Datenstruktur, die beginnend mit dem Wurzelknoten einzelne Tasks, aber auch weitere Task Trees besitzen kann.

Task

Ein Task (engl. für Aufgabe) beschreibt einen Arbeitsschritt innerhalb eines wissensintensiven Prozesses. Ein Task kann ein To-do oder ein Checkpunkt in entsprechenden Task Trees sein und stellt Wissensarbeitern wichtige Informationen, die zur Bearbeitung der Aufgabe notwendig sind, bereit. So besitzt ein Task immer eine Aufgabenbeschreibung und den aktuellen Bearbeitungsstatus.

Templates und Instanzen

Um den Lebenszyklus eines wissensintensiven Prozesses abzubilden, können die drei Schlüsselkomponenten Prozesse, Task Tree und Task als Vorlagen und Instanzen angelegt werden. Als wichtigstes Element kann eine Prozessvorlage (engl. Process Template) mit beliebig vielen *Task Tree Templates* verknüpft sein, welche ihrerseits *Task Templates* und weitere *Sub Task Tree Templates* enthalten können. Dies gilt analog auch für Prozess-, Task Tree- und Taskinstanzen.

Ein Template ist vergleichbar mit einer Vorlage zur Verwirklichung einer Grundidee, es soll Wissensarbeiter bei der Planung von wissensintensiven Prozessen unterstützen und diese beschleunigen [MR17]. Wissensarbeiter können ein Prozess Template auswählen, das am besten ihren Anforderungen an den wissensintensiven Prozess gerecht wird und dieses dann instanziieren. Ist kein passendes Template vorhanden, kann eine neue Prozessinstanz auch ohne Template erstellt werden. Eine Prozessinstanz kann so ein laufendes Projekt oder ein andere Art der Zusammenarbeit (z.B. Fälle, Ermittlungen, Prüfungen) zwischen Wissensarbeitern darstellen. Mit einer Prozessinstanz werden auch alle untergeordneten Elemente (Task Trees und Tasks) instanziiert und dazugehörige Eigenschaften und Ressourcen (z.B. Datenelemente) kopiert. Darüber hinaus kann eine Prozessinstanz beliebig viele Task Tree Instanzen und Sub Task Tree Instanzen enthalten.

Um den Charakteristiken von wissensintensiven Prozessen gerecht werden zu können, ist es Wissensarbeitern jederzeit möglich, die Prozessinstanzen während der Laufzeit beliebig auf die aktuellen Begebenheiten anzupassen und angepasste Prozessinstanzen für zukünftige Prozesse als Template abzuspeichern. So entstehen über die Zeit immer stärker an konkrete wissensintensive Prozesse angepasste Task Trees und Prozesse.

2.2.3 Workspace

Die drei Schlüsselkomponenten Prozesse, Task Trees und Tasks werden in Workspaces verwaltet. In proCollab können beliebig viele Workspaces erstellt und verwaltet werden, dadurch kann für jeden Kunden des proCollab-Systems ein eigener Workspace angelegt werden. Zum Beispiel befinden sich in einem Workspace all jene Projekte eines Unternehmens, bei denen es um die Entwicklung von Websites geht, während in einem anderen alle Entwicklungen für mobile Applikationen eines anderen Unternehmens gehalten werden.

2.2.4 Datenmanagement in proCollab

Für das Datenmanagement steht in proCollab ein Data Repository pro Workspace zur Verfügung. Das Data Repository ermöglicht eine persistente Organisation und Speicherung der Datenelemente. Dazu werden die Datenelemente auch in die bestehende Systemstruktur eingebunden, das heißt es ist möglich, Datenelemente mit proCollab Schlüsselkomponenten zu verbinden (siehe Abbildung 2.3).

Datenelemente

Unter dem einheitlichen Begriff Datenelemente werden in proCollab zwei verschiedene Arten von Daten unterschieden: *Unstrukturierte* und *strukturierte Daten*.

Unstrukturierte Daten sind Dateien, denen keine vordefinierte Struktur zugrunde liegt. Dazu gehören zum Beispiel Text- und PDF-Dateien, aber auch Bild-, Audio- und Videodateien.

Bei strukturierten Daten handelt es sich um Daten denen ein Schema (*strukturiertes Daten Template*) zugrunde liegt. So kann eine Adresse ein strukturiertes Datum sein, das die Attribute Straße, Hausnummer, Postleitzahl und Ort enthält.

Data Repository und Data Spaces

Das *Data Repository* realisiert das Datenmanagement in proCollab und verwaltet die *Data Spaces* (siehe Abbildung 2.3). Um Datenelemente aufnehmen zu können, sind die Systemobjekte Prozess Template und Instanz, Task Tree Template und Instanz, sowie

der Workspace mit je einem eigenen Data Space verbunden, der die Datenelemente der Komponenten verwaltet. Um unstrukturierte Daten besser organisieren zu können, besteht zusätzlich die Möglichkeit, in einem Data Space Ordner anzulegen und darin unstrukturierte Daten und weitere Ordner abzulegen.

Ein Task Template und eine Taskinstanz besitzen keinen eigenen Data Space. Dafür kann hier auf Datenelemente aus anderen Data Spaces verwiesen werden und diese als *Input*, *Output* oder als *Hybrid* festgelegt werden. Dabei definiert der Input die für den Start des Tasks benötigen Datenelemente, während der Output das Ergebnis darstellt. Als Hybrid markierte Datenelemente sind sowohl als Input, als auch als Output zuzuordnen.

Eine weitere Funktion, die Schlüsselkomponenten durch das Data Repository bereitgestellt wird, ist das Verlinken von Datenelementen aus anderen Data Spaces. Allerdings können die Schlüsselkomponenten nur Datenelemente aus dem Data Space des Workspaces oder aus den Data Spaces von Schlüsselkomponenten denen sie selbst angehören verlinken. So kann eine Taskinstanz auf ein Datenelement im Data Space des Task Trees, der Prozessinstanz sowie dem Workspace, denen sie zugeordnet ist, verlinken. Umgekehrt kann aus dem Data Space einer Prozessinstanz allerdings nicht auf ein Datenelement im Data Space eines ihr untergeordneter Taskinstanz verwiesen werden. Eine weitere Ausnahme bildet der Data Space auf Workspace-Ebene: Hier ist es nicht möglich strukturierte Daten abzulegen und, da es die höchste Ebene ist, können auch keine verlinkten Datenelemente hinterlegt werden.

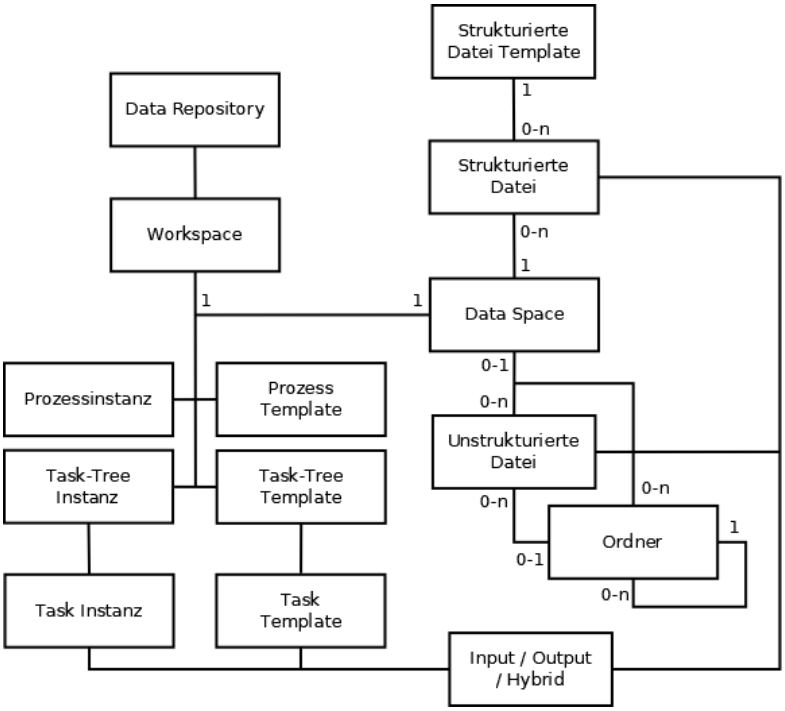


Abbildung 2.3: proCollab Datenmanagement

2.3 Usability

Usability ist ein allgemeiner Begriff der besagt, dass etwas so funktionieren soll wie es angedacht ist und von den Benutzern für die diese Sache entwickelt wurde, in der beabsichtigten Weise benutzt werden kann ohne dabei zu frustrieren [May10]. Der Standard ISO 9241-11 [DIN16] mit dem Titel „Ergonomie der Mensch-System-Interaktion“ definiert Usability wie folgt:

> *„Usability gibt den Grad an zu welchem ein Produkt durch einen Nutzer benutzt werden kann um seine Ziele mit Effektivität, Effizienz und Zufriedenheit im Nutzungskontext zu erreichen.“*

Effektivität beschreibt die Genauigkeit und Vollständigkeit, mit der Benutzer bestimmte Ziele erreichen. Effizienz gibt die von Benutzern aufgewendeten Mittel im Verhältnis zur Genauigkeit und Vollständigkeit, mit der Benutzer ihre Ziele erreichen an. Zufriedenheit beschreibt das Wohlbefinden und die positive Einstellung gegenüber der Nutzung des Produkts [DIN16].

Um einen gewissen Grad an Usability in der Entwicklung des UI-Konzepts zu gewährleisten, werden die im nächsten Unterkapitel aufgeführten Richtlinien herangezogen.

2.3.1 Grundsätze der Dialoggestaltung

Die Grundsätze der Dialoggestaltung gelten als allgemeine Leitlinien zur Dialoggestalltung. Nicht jeder Grundsatz kann zum gleichen Maße angewendet werden. Es muss bei der Entwicklung abgewogen werden, ob und wenn ja, wie stark ein Grundsatz bei der Dialoggestaltung genutzt werden kann. Die Grundsätze der Dialoggestaltung werden in EN ISO 9241, Teil 110 [DIN16] wie folgt definiert:

Aufgabenangemessenheit

> *„Ein Dialog ist aufgabenangemessen, wenn er den Benutzer unterstützt, seine Arbeitsaufgabe effektiv und effizient zu erledigen.“*

Um die Aufgabenangemessenheit zu gewährleisten werden in der Aufgabenanalyse in Kapitel 3.2 die Aufgaben identifiziert, die der Benutzer am System durchführen können muss.

Selbstbeschreibungsfähigkeit

> *„Ein Dialog ist selbstbeschreibungsfähig, wenn jeder einzelne Dialogschritt durch Rückmeldung des Dialogsystems unmittelbar verständlich ist oder dem Benutzer auf Anfrage erklärt wird.“*

Dem Benutzer sollte jederzeit klar sein, wo er sich im System befindet, wie er dort hin kam, was er dort machen kann und wohin er von hier aus gelangen kann.

Erwartungskonformität

> *„Ein Dialog ist erwartungskonform, wenn er konsistent ist und den Merkmalen des Benutzers entspricht, z.B. den Kenntnissen aus dem Arbeitsgebiet, der Ausbildung und der Erfahrung des Benutzers sowie den allgemein anerkannten Konventionen.“*

Es ist eine konsistente Dialoggestaltung wichtig, die den bisherigen Erfahrungen die der Benutzer mit dem System verbindet, entspricht. Deshalb sollte ein einheitliches Verhalten der Dialoge des Systems vorhanden sein und für ähnliche Aufgaben sollten auch die Dialoge ähnlich gestaltet sein.

Fehlertoleranz

> *„Ein Dialog ist fehlertolerant, wenn das beabsichtigte Arbeitsergebnis trotz erkennbar fehlerhafter Eingaben entweder mit keinem oder mit minimalem Korrekturaufwand seitens des Benutzers erreicht werden kann.“*

Es stellt sich die Frage wie sich das System bei Eintritt eines Fehlers verhält. Entscheidend ist dabei welche Maßnahmen zur Fehlervermeidung und Behebung unternommen werden können und wie Fehlermeldungen dargestellt werden [Off16].

Lernförderlichkeit

> *„Ein Dialog ist lernförderlich, wenn er den Benutzer beim Erlernen des Dialogsystems unterstützt und anleitet.“*

Um die Lernförderlichkeit zu erhöhen, kann sich bei der Dialoggestaltung an dem für den Benutzer bekannten und gewohnten Mustern aus ähnlichen Anwendungen orientiert werden. Es gilt darauf zu achten, den Benutzer nicht mit komplexen Dialogen zu überfordern, aber dennoch alle relevanten Informationen zu liefern.

Individualisierbarkeit

> *„Ein Dialog ist individualisierbar, wenn das Dialogsystem Anpassungen an die Erfordernisse der Arbeitsaufgabe, individuelle Vorlieben des Benutzers und Benutzerfähigkeiten zulässt.“*

Dazu können Anpassungen wie die Veränderung der Schriftgröße, das Anzeigen alternativer Dialogdarstellungen oder das Einblenden zusätzlicher Informationen gehören. Dabei gilt es allerdings zu beachten, den Benutzer nicht mit zu vielen Einstellungsmöglichkeiten zu überfordern.

Steuerbarkeit

> *„Ein Dialog ist steuerbar, wenn der Benutzer in der Lage ist, den Dialogablauf zu starten sowie seine Richtung und Geschwindigkeit zu beeinflussen, bis das Ziel erreicht ist.“*

Einige wichtige Punkte, die Einfluss auf die Steuerbarkeit eines Dialogablaufs haben, sind Abbruchmöglichkeiten und die Möglichkeit jederzeit zur Startseite zurückzukehren, sowie das Verhalten des Systems in kritischen Situationen.

3

Anforderungsanalyse

Mit der Anforderungsanalyse sollen Informationen gewonnen werden, die die Grundanforderungen an das zu entwickelnde UI-Konzept beschreiben. Dazu wird zuerst der Ist-Zustand (siehe Kapitel 3.1.1) des proCollab-Prototypen festgehalten. Darauf folgt ein Vergleich mit proCollab ähnlichen Systemen (siehe Kapitel 3.1.2) um herauszufinden, wie diese Datenmanagement in ihr System integrieren. Die Aufgabenanalyse (siehe Kapitel 3.2) definiert die Anforderungen an das Konzept. Zuletzt wird in Kapitel 3.3 der Entwicklungskontext betrachtet und identifiziert für welche Zielsysteme und welche Auflösung das Konzept entwickelt werden soll.

3.1 Ist-Zustand

Das Ziel ist es den aktuellen Systemzustand von proCollab, im Hinblick auf das Datenmanagement, zu ermitteln. Das Ergebnis bildet dann die Grundlage für die nachfolgenden Schritte der Anforderungsanalyse und der Entwicklung des Konzepts. Der Vergleich ähnlicher Systeme soll bei der Ideensammlung für das UI-Konzept helfen.

3.1.1 proCollab

Bereits in Kapitel 2 wurde die Grundidee und der Aufbau von proCollab erläutert. Nun werden die für diese Arbeit relevanten Systemkomponenten genauer betrachtet, um den derzeitigen Stand des Systems als Grundlage für die weitere Analyse zu bilden. Das Projekt proCollab besteht aus einer serverseitigen Implementierung und einem Web-Client. Der Server stellt die Programmlogik und Datenbank bereit, während der Web-Client die Benutzeroberfläche darstellt und Interaktion mit den Nutzern ermöglicht. Im Folgenden werden diese beiden Hauptbestandteile von proCollab getrennt vonein-

ander betrachtet, wobei auf der Serverseite nur das für diese Arbeit notwendige Data Repository angeschaut wird.

Web-Client

Die Hauptansicht des Web-Clients (siehe Abbildung 3.1) kann in drei Bereiche eingeteilt werden. Am linken Rand befindet sich eine Navigationsleiste, worüber in die entsprechenden Menüpunkte gesprungen werden kann. Rechts daneben, grau hinterlegt, befindet sich der Inhaltsbereich. Er ist flächenmäßig der größte Bereich und wird dazu genutzt die Inhalte des ausgewählten Menüpunktes anzuzeigen. Über dem Inhaltsbereich ist eine Brotkrümelnavigation angebracht. Mit ihrer Hilfe kann der Benutzer erkennen wo er sich gerade im System befindet und über welche Wege er dort hin gelangt ist, aber auch an jeden vorherigen Punkt zurück springen.

Im aktuellen Zustand lassen sich im Web-Client Prozess-Templates, Prozessinstanzen, Task Trees und Tasks angelegen und bearbeiten. Ist eine Prozessinstanz oder ein Prozess Template geöffnet, wird der Navigationsleiste der Menüpunkt „Data and Documents“ hinzugefügt. Dies ist ein zentraler Ort zum Betrachten der in der Prozessinstanz oder dem Prozess Template hinterlegten Datenelemente. Der Inhaltsbereich ist in drei Spalten unterteilt. In der ersten Spalte werden die prospektiven Tasks (z.B. als To-do Liste mit den dazugehörenden To-dos) angezeigt, in der rechten Spalte befinden sich die retrospektiven Tasks (z.B. als Checkliste mit den dazugehörenden Checkpunkten). Die mittlere Spalte gewährt einen Projektüberblick. In Dropdown-Boxen werden hier Projekteigenschaften wie Projektziel oder Startzeitpunkt angezeigt. Eine Dropdown-Box gewährt einen Überblick über vorhandene Unterprojekte, während die Dropdown-Box „Data and Documents“, zur Anzeige der dem Projekt zuletzt hinzugefügten Datenelementen bisher noch ohne Funktion ist.

Wird eine To-do Liste oder eine Checkliste ausgewählt, öffnet sich der gleichnamige Menüpunkt und die Liste nimmt den gesamten Inhaltsbereich ein. In der linken Spalte des Inhaltsbereichs befindet sich die To-do Liste oder Checkliste und rechts davon die To-dos oder Checkpunkte der Liste. Jede Liste und jedes ihrer Elemente lässt sich über einen Dropdown-Button bearbeiten, löschen oder durch neue Elemente erweitern.

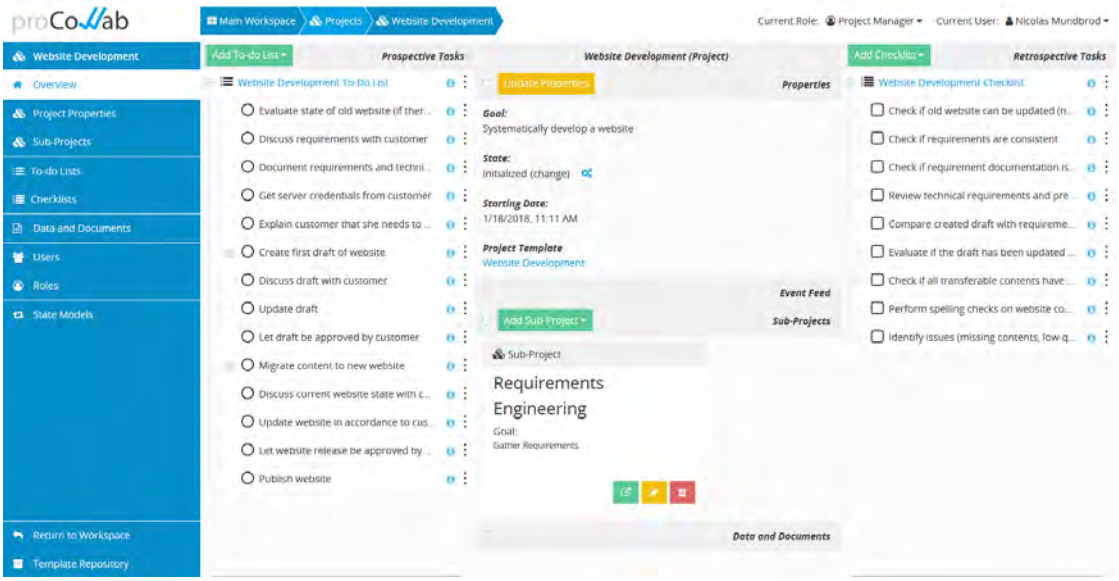


Abbildung 3.1: Hauptansicht des proCollab Web-Client bei geöffneter Projektinstanz

Server

Im proCollab Server ist das Data-Repository, das die Funktionen für die Verwaltung der Datenelemente bereitstellt, implementiert. So können unter anderem unstrukturierte Daten hochgeladen, umbenannt, verlinkt und auch wieder gelöscht werden.

Einem Task können Datenelemente als Input, Output, bzw. Hybrid zugewiesen und auch wieder entfernt werden. Da beim Festlegen der Startbedingung und vor allem der Endvoraussetzung noch nicht unbedingt das entsprechende Datenelement zur Verfügung steht, können Platzhalter definiert werden. Dieser Platzhalter legt Dateinamen und Dateityp fest, die das noch benötigte Datenelement besitzen muss um den Task starten oder abschließen zu können.

Der Server ermöglicht das Erstellen strukturierter Daten aus bestehenden strukturierten Daten Templates. Das Erstellen, Bearbeiten und Löschen von Templates wird ebenfalls unterstützt. Strukturierte Daten Templates benötigen einen Bezeichner und können mehrere Attribute, bestehend aus Bezeichner und Datentyp, enthalten, die diese strukturierte Datei genauer beschreiben. Jedes strukturierte Daten Template kann geändert oder gelöscht werden, wobei eine Änderung nur dann vorgenommen werden kann, wenn das Template nicht schon als strukturierte Datei innerhalb einer proCollab Systemkomponente genutzt wird.

3.1.2 Vergleichbare Systeme

Um einen Eindruck von proCollab ähnelnden Systemen, die schon auf dem Markt existieren, zu gewinnen, wird nachfolgend das Datenmanagement von vier vergleichbaren Systemen verglichen. Dabei wird festgehalten, wo Daten abgelegt werden können, wie diese dargestellt werden und welche Informationen über die hinterlegten Daten ersichtlich sind. Die Vergleiche dienen dem Ziel Lösungsansätze zu finden, um daraus Ideen für das zu entwickelnde UI-Konzept zu erhalten, aber auch um eventuelle Fehler zu vermeiden.

In den folgenden Beschreibungen werden unstrukturierte Daten als Dateien bezeichnet, strukturierte Daten werden von keinem diese Systeme unterstützt.

Asana

Asana [Asa17] ermöglicht das Einbinden von Dateien in Task Listen, Tasks, Subtasks und Kommentaren aus lokaler Quelle über einen Dateiselektor und Drag & Drop oder externe Dienstleister (siehe Abbildung 3.2). Dargestellt werden die Dateien als einfache Liste unterhalb der Taskbeschreibung. Der Dateiname ist die einzige Information zu einer Datei, eine Beschreibung muss in der jeweiligen Taskbeschreibung erfolgen. Projekt-Templates werden von Asana unterstützt, allerdings ist diese Funktion in der kostenlosen Version stark eingeschränkt. Eine genaue Betrachtung dieser Funktion ist dadurch nicht möglich.

Alle im Projekt hinterlegten Daten werden im Tab „Files“ als Thumbnail in einem Raster angezeigt. Als zusätzliche Information wird neben dem Dateinamen, der Ersteller und der Ablageort der Datei genannt. Durch Klick auf ein Thumbnail wird der Task oder der Kommentar, an dem die Datei hinterlegt wurde, geöffnet. Haben zwei Dateien einen unterschiedlichen Ursprungsort, aber denselben Dateinamen, ist der genaue Ursprungsort nur durch einen Klick auf die Dateien und des sich dadurch öffnenden Ursprungsortes, eindeutig herauszufinden. Es gibt weder Filter noch sonstige Anzeige- oder Sortierfunktionen. Ein direkter Upload in diese Übersicht ist ebenfalls nicht möglich.

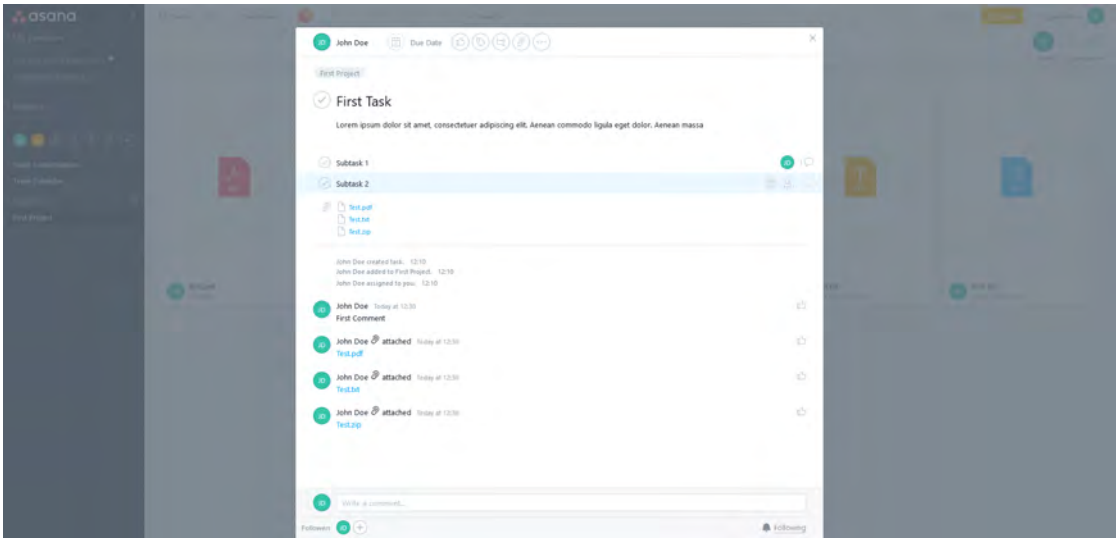


Abbildung 3.2: Asana

Basecamp

In Basecamp [Bas17] können Dateien aus lokaler Quelle an To-do Listen, Tasks, Nachrichten und Kommentare angehängt werden. Dargestellt werden alle Dateien durch Thumbnails mit Dateinamen und Dateigröße als zusätzliche Information. Diese Darstellung lässt sich nicht ändern, es besteht allerdings die Möglichkeit angehängte Dateien zu verbergen, um so die Übersichtlichkeit zu erhöhen.

Eine Besonderheit von Basecamp ist die „Docs & Files“-Sektion (siehe Abbildung 3.3), die Teil eines jeden Projekts ist und im Vergleich zu Asana und Wrike erweiterte Möglichkeiten zur Organisation von Daten bereitstellt. Hier können Dateien aus einer lokalen Quelle hochgeladen und beliebig umbenannt werden. Zusätzlich kann zu jeder Datei noch eine Notiz hinzugefügt werden. Weiter besteht die Möglichkeit ein Textdokument mit einfachen Formatierungen und Anhang aus einer lokalen Quelle zu erstellen. Alternativ kann auch ein Google Doc Dokument erstellt und eingebunden werden, das zusätzlich noch mit einem Titel und einer optionalen Notiz versehen werden kann. Für eine erhöhte Übersichtlichkeit sorgt das Anlegen von Ordnern, in denen beliebige Dateien und weitere Ordner abgelegt werden können. Per Drag & Drop ist es möglich Ordner und Dateien in andere Ordner zu verschieben.

Alle Dateien lassen sich durch andere (z.B. neuere Versionen) ersetzen. Durch eine Versionierung ist es jederzeit möglich, zu einer älteren Version der Datei zurückzuspringen oder sich die Änderungen anzeigen zu lassen.

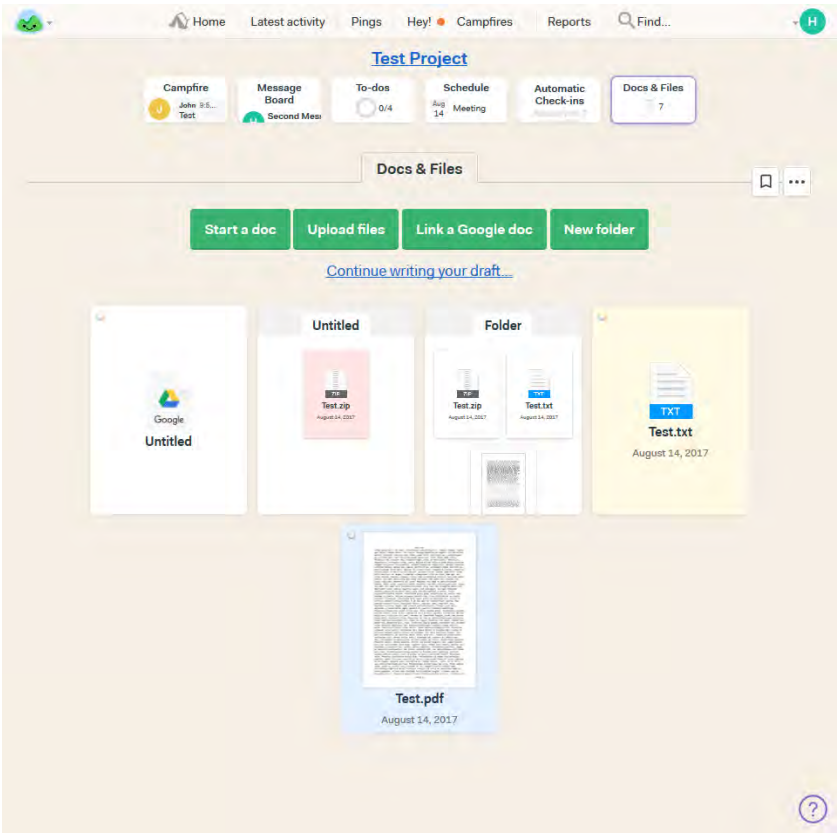


Abbildung 3.3: Docs & Files von Basecamp

Wrike

Wrike [Wri17] ist weniger kommunikationsorientiert als die bisher betrachteten Systeme, bietet dafür aber umfangreichere Bearbeitungsmöglichkeiten der Dateien. Dateien können als Anhang an die Projektbeschreibung, Tasks, Subtasks und Kommentare aus lokaler Quelle oder externem Dienstleister integriert werden. Dargestellt werden diese als Thumbnails mit Dateinamen, Uploaddatum und Name des Erstellers (siehe Abbildung 3.4).

An Projektbeschreibung, Tasks und Subtasks angehängte Dateien können einem Nutzer zur Begutachtung zugewiesen werden. Diese Dateien werden optisch getrennt von den restlichen Anhängen, die nicht zur Begutachtung markiert wurden, dargestellt. Alle angehängte Dateien können nach Datum oder Dateinamen sortiert werden. Wie in Basecamp unterstützt auch Wrike die Versionierung von Dateien. Nutzer haben die Möglichkeit eine Datei durch eine neuere Version zu ersetzen. Die neue Datei wird automatisch umbenannt, um der schon existierenden zu entsprechen und es wird ein

Versionshinweis hinzugefügt (v1, v2, usw.). Es findet zusätzlich eine Überprüfung des Dateiformates statt, denn Dateien dürfen nur durch Dateien gleichen Formats ersetzt werden.

Wrike bietet eine Editierfunktion für angehängte Dokumente, die automatisch die Datei im Standardeditor des Systems für diesen Dateityp öffnet. Wird die editierte Datei anschließend abgespeichert, wird sie als neue Version in das Projekt hochgeladen. Als einziges der bis jetzt verglichenen Systeme enthält Wrike keine zentrale Datenübersicht.

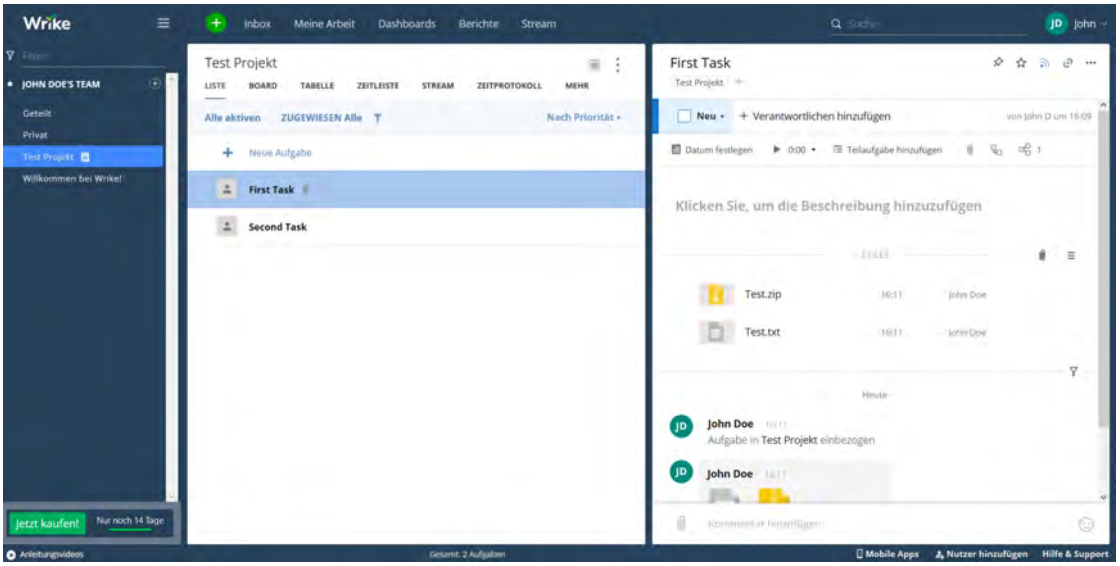


Abbildung 3.4: Wrike

Dropbox

Bei Dropbox [Dro17] handelt es sich zwar nicht um ein Aufgabenmanagement System, dafür bietet es ein umfangreiches Datenmanagement und ist deshalb in diesem Zusammenhang ebenfalls von Interesse. Dateien können aus lokaler Quelle hochgeladen werden und werden anschließend in einer Liste angezeigt (siehe Abbildung 3.5). Auch das Anlegen von Ordnern wird unterstützt. Als alternative Ansicht kann zu einer Thumbnail-Ansicht gewechselt werden, die Dateien und Ordner in einem Raster anzeigt. Unabhängig von der gewählten Ansicht wird immer nahezu der gesamte Bildschirm für

die Darstellung der Daten genutzt. Neben dem Dateinamen werden noch das Änderungsdatum, Dateityp, Endung und Größe angezeigt. Zudem lassen sich die Dateien nach jeder dieser Eigenschaften auf- oder absteigend sortieren.

Alle Dateien und Ordner lassen sich umbenennen, verschieben und kopieren. Das Verschieben und Kopieren der Dateien erfolgt entweder durch Drag & Drop, oder über einen Dialog von welchem das gewünschte Verzeichnis ausgewählt werden kann. Eine Bestätigung startet anschließend den Vorgang.

Gelöschte Dateien können in der kostenfreien innerhalb von 30 und in der kostenpflichtigen Version innerhalb von 120 Tagen wiederhergestellt werden. Über eine Versionierung kann jede Änderung an einer Datei eingesehen und rückgängig gemacht werden.

Abbildung 3.5: Dropbox

Erkenntnisse

Es fällt bei Asana und Wrike auf, dass Datenmanagement keine zentrale Rolle einnimmt, zwar können an Task Trees und Task Daten eingebunden werden, doch sind diese auf sehr begrenzten Raum zwischen Aufgabenbeschreibung und Kommentaren untergebracht. Dies mindert die Übersichtlichkeit und lässt keinen Platz für zusätzliche Informationen über die Daten. Besser wird das von Basecamp und Dropbox gelöst. Hier widmet sich die gesamte Ansicht den Daten und bietet so ausreichend Raum um diese übersichtlich darzustellen. Zusätzlich ist es bei beiden möglich Dateien in Ordnern zu organisieren, um so für noch mehr Übersichtlichkeit zu sorgen.

Als reines Datenmanagement System bietet Dropbox viele Darstellungen die auch für das proCollab UI-Konzept interessant sind. So werden Dateien und Ordner in einer Liste angezeigt, die zusätzlich über Spalten verfügt, die weitere Informationen über die

Dateien liefert und es ermöglicht die Dateien nach diesen Informationen zu sortieren. Für das Verschieben von Dateien bietet Dropbox, neben Drag & Drop, die Möglichkeit über einen separaten Dialog den Zielort aus einem Verzeichnisbaum auszuwählen und so komfortabel Dateien zu verschieben.

Asana bietet zwar eine Übersicht über alle im Projekt hinterlegten Dateien, doch ist die Herkunft nicht immer eindeutig zu erkennen. Dies gilt es bei dem Entwurf des proCollab UI-Konzepts zu berücksichtigen. Der Benutzer muss beim Betrachten von Datenelementen sofort erkennen können wo diese hinterlegt sind.

Keine Erkenntnisse liefern die hier verglichenen Systeme über die Darstellung strukturierter Daten, sowie der Möglichkeit Daten als Input oder Output eines Task Trees oder Tasks festzulegen.

3.2 Aufgabenanalyse

In diesem Kapitel werden die Aufgaben festgehalten, die der Benutzer am Ende mit dem System erfüllen können soll. Die Anwendungsfälle werden zuerst in UML Anwendungsfalldiagramme gesammelt und schematisch dargestellt. Anschließend wird jeder Anwendungsfall genauer beschrieben und die Interaktionen zwischen den Nutzern und dem System für jeden dieser Fälle in einem Flussdiagramm (nach ISO 5807) genau identifiziert. So soll festgehalten werden, welche Eingaben der Benutzer am System vornehmen muss und welche Informationen und Operationen das System dem Benutzer zur Verfügung stellen muss um die jeweilige Aufgabe erfolgreich durchführen zu können. Die Analyse dieser Anwendungsfälle im Detail bilden anschließend die Basis für die in Kapitel 4 folgende UI-Konzeptphase.

Elemente eines Flussdiagramms

Für die Flussdiagramme werden die in Tabelle 3.1 dargestellten Elemente verwendet.

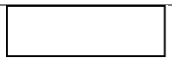
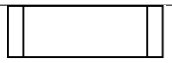
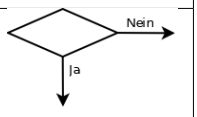
Start / Stop	
Tätigkeit	
Vordefinierter Prozess	
Flussrichtungspfeil	
Entscheidung	
Akteur	

Tabelle 3.1: Elemente eines Flussdiagramms

3.2.1 Datenmanagement von Data Spaces

In proCollab können Datenelemente auf Workspace-Ebene, in Prozess Templates, Prozessinstanzen und den dazugehörigen Task Tree Templates, Task Tree Instanzen hinterlegt werden. Es folgen Operationen, die von allen Systemnutzern über die Benutzerschnittstelle an diesen Data Spaces durchgeführt werden können (siehe Abbildung 3.6).

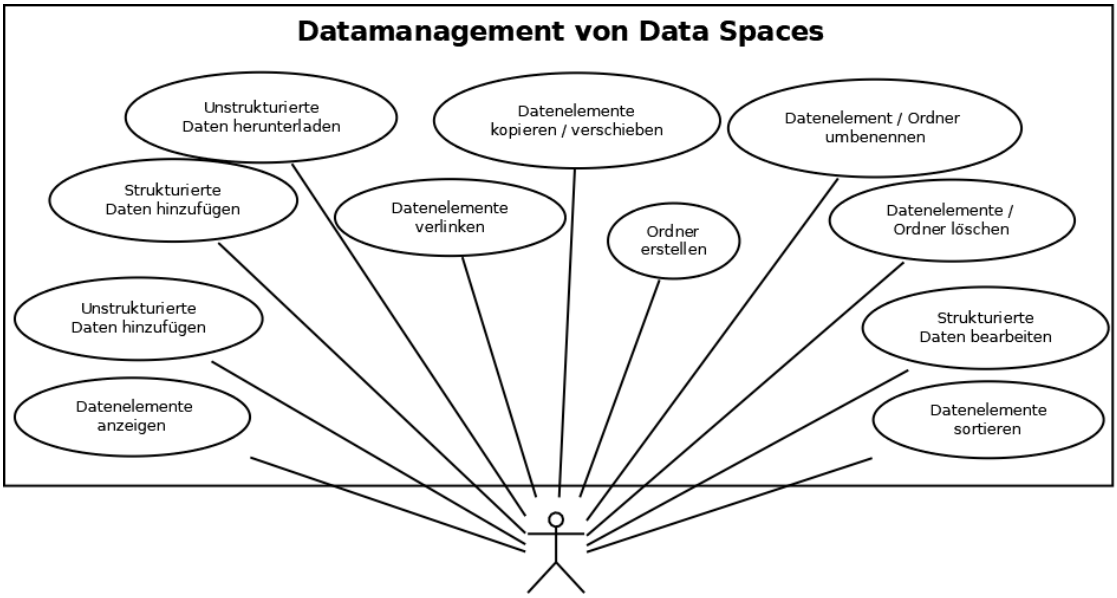


Abbildung 3.6: Use-Case Diagramm zum Datenmanagement von Data Spaces

Datenelemente anzeigen

Sind in einem Data Space Datenelemente hinterlegt, kann sich der Benutzer diese anzeigen lassen und erhält so weitere Informationen z.B. Dateinamen und Dateityp. Erst wenn Datenelemente angezeigt werden, kann mit ihnen interagiert werden, um z.B Datenelemente zu löschen oder zu verschieben (siehe Abbildung 3.7).

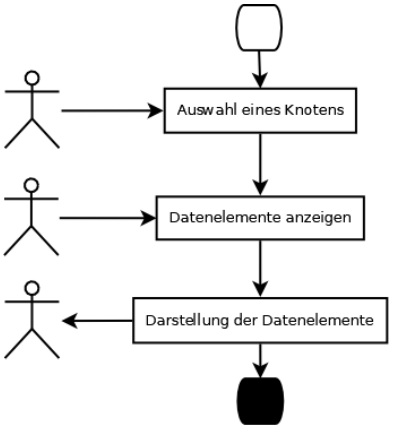


Abbildung 3.7: Flussdiagramm: Datenelemente anzeigen

Unstrukturierte Daten hinzufügen

Jedem Data Space können beliebig viele Datenelemente hinzugefügt werden. Unstrukturierte Daten sollen mit einem Dateiselektor von einer lokalen Quelle (z.B. Festplatte des Computers auf dem der Client läuft) in den ausgewählten Data Space hinzugefügt werden können (siehe Abbildung 3.8).

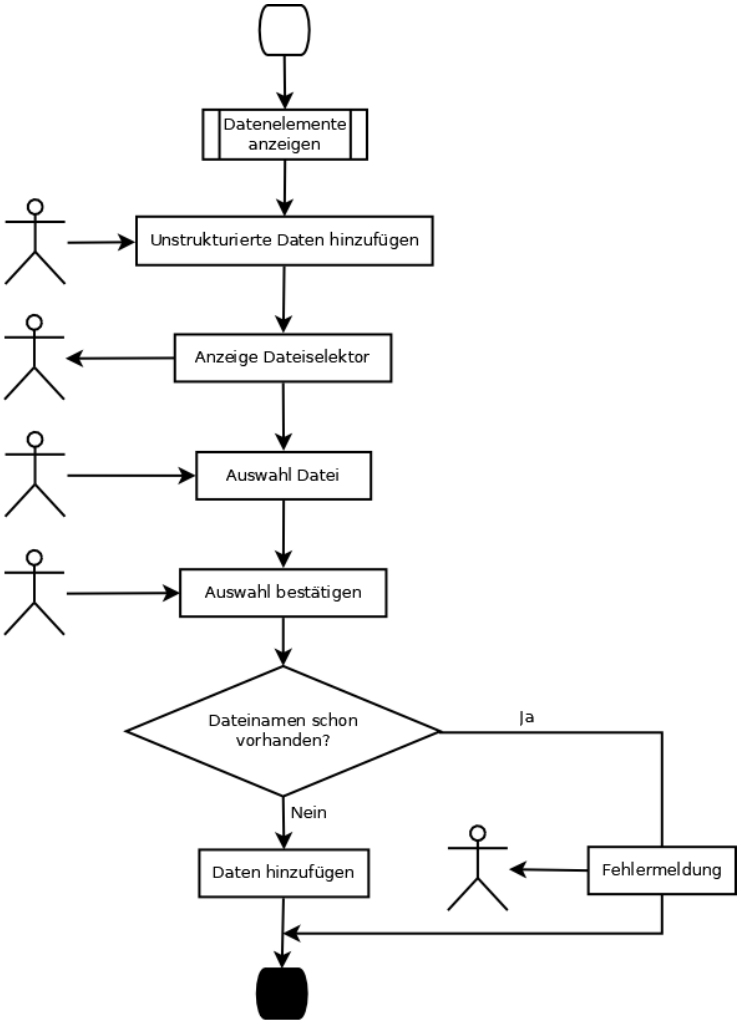


Abbildung 3.8: Flussdiagramm: Unstrukturierte Daten hinzufügen

Strukturierte Daten hinzufügen

Strukturierte Daten können nur dann hinzugefügt werden, wenn entsprechende Daten Templates davon existieren (siehe Kapitel 3.2.3). Existieren solche Daten Templates kann der Nutzer aus einer Auflistung das gewünschte auswählen, einen Bezeichner vergeben und es dem Data Space hinzufügen (siehe Abbildung 3.9).

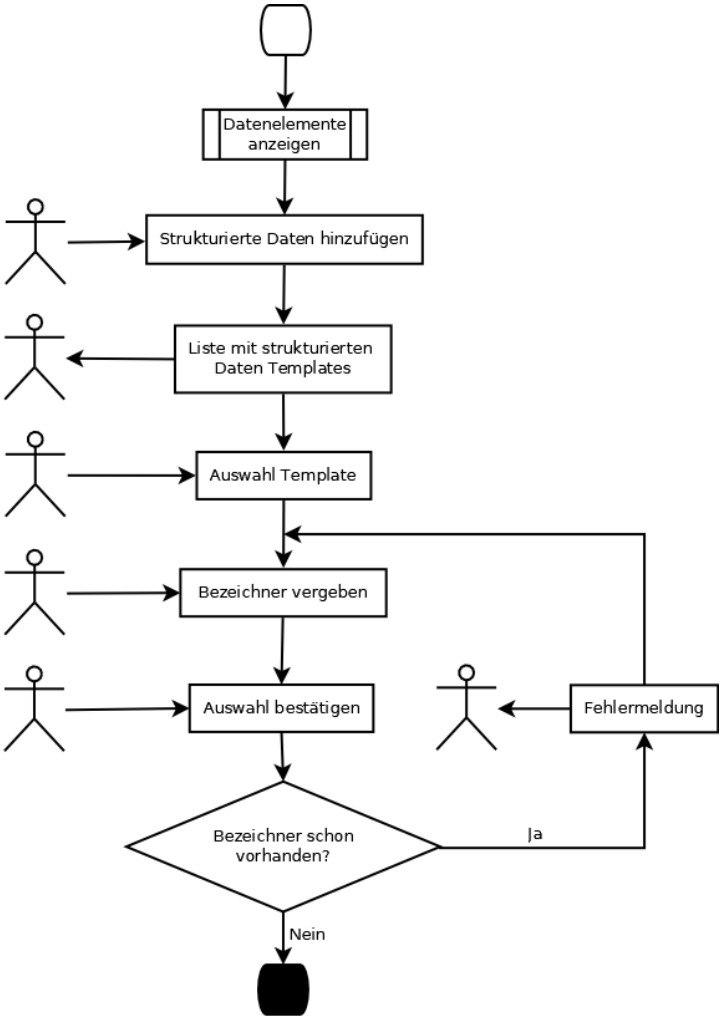


Abbildung 3.9: Flussdiagramm: Strukturierte Daten hinzufügen

Datenelement verlinken

proCollab unterstützt die Verlinkung von Datenelementen zwischen den Data Spaces. Dazu kann einem Data Space aus einem in der Hierarchie gleich oder höher gestellten Data Space Datenelemente verlinkt werden. Der Nutzer selektiert den gewünschten Data Space und wählt anschließend das gewünschte Datenelement und bestätigt die Verlinkung. Das verlinkte Datenelement erscheint dann an dem Ort, an dem der Vorgang gestartet wurde (siehe Abbildung 3.10).

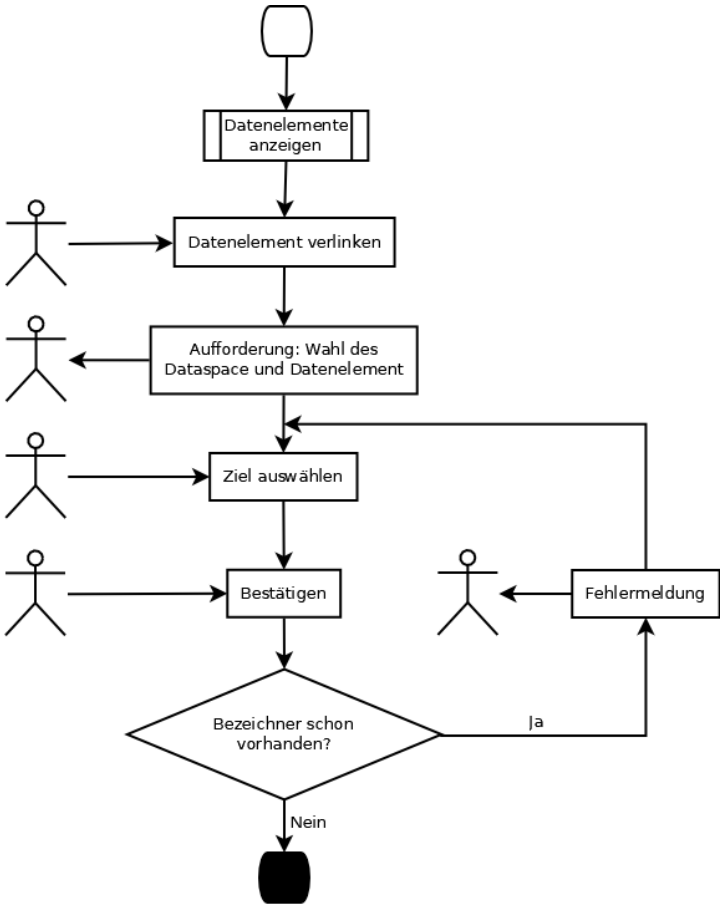


Abbildung 3.10: Flussdiagramm: Datenelement verlinken

Datenelemente kopieren / verschieben

Zur einfacheren Handhabung der Datenelemente können diese durch Kopieren oder Verschieben zwischen den Data Spaces bewegt werden. Nach dem Markieren der zu kopierenden oder zu verschiebenden Datenelemente, kann der Benutzer den gewünschte Data Space auswählen und darin zum genauen Ziel navigieren und den Vorgang durchführen (siehe Abbildung 3.11).

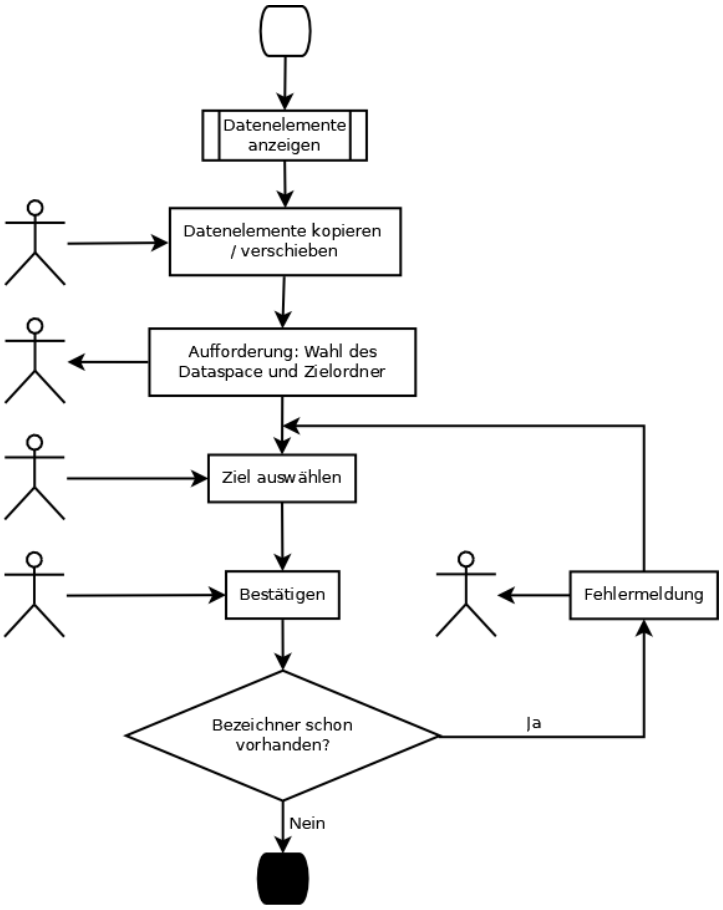


Abbildung 3.11: Flussdiagramm: Datenelemente kopieren / verschieben

Datenelemente sortieren

Die Übersicht über die Datenelemente soll mit einer Sortierfunktion erweitert werden. Damit lassen sich Datenelemente nach gemeinsamen Eigenschaften gruppieren sowie auf- oder absteigend anordnen (siehe Abbildung 3.12).

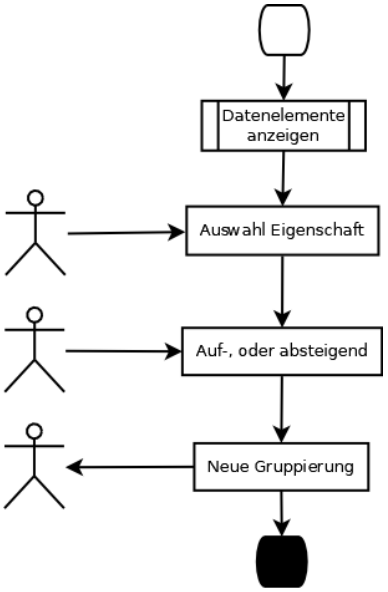


Abbildung 3.12: Flussdiagramm: Datenelemente sortieren

Unstrukturierte Daten herunterladen

Werden in einem Data Space Datenelemente angezeigt, können diese direkt von dort heruntergeladen werden. Im „Speichern unter“-Dialog des Betreibsystems, auf dem der Web-Client ausgeführt wird, kann der Nutzer den Speicherort selbst auswählen (siehe Abbildung 3.13).

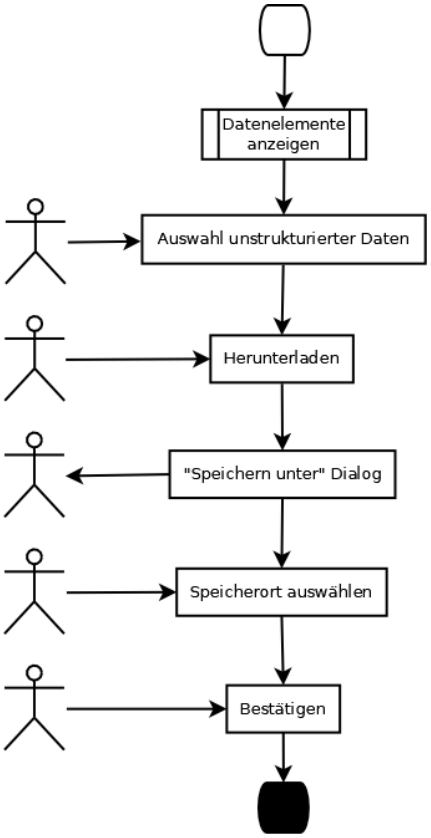


Abbildung 3.13: Flussdiagramm: Unstrukturierte Daten herunterladen

Strukturierte Daten bearbeiten

Über einen Button können strukturierte Daten eines Data Spaces bearbeitet werden. Die Bezeichner der strukturierten Daten können nicht geändert werden, Änderungen können nur an den Attributswerten vorgenommen werden, dabei ist auf den Datentyp zu achten. Entspricht die Eingabe des Benutzers nicht dem benötigten Datentyp, wird durch auf einen Hinweis darauf aufmerksam gemacht und auf den Datentyp hingewiesen (siehe Abbildung 3.14).

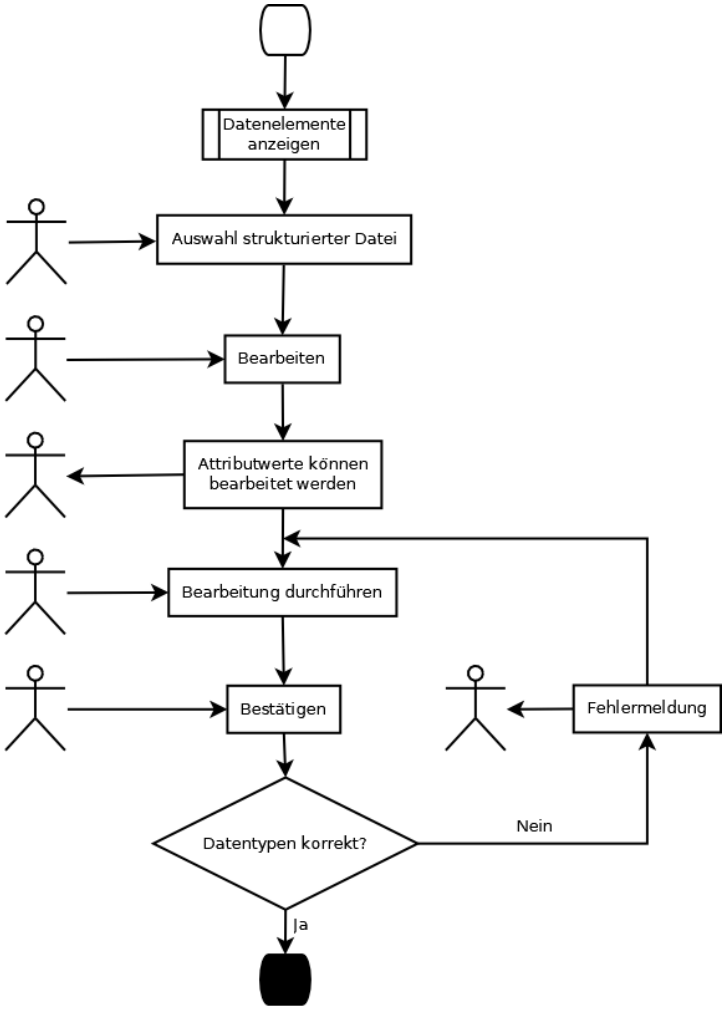


Abbildung 3.14: Flussdiagramm: Strukturierte Daten bearbeiten

Ordner erstellen

Damit die Übersichtlichkeit der Datenelemente eines Knotens gewährleistet werden kann, besteht die Möglichkeit diese in Ordnern zu organisieren. In einem Ordner dürfen sich beliebig viele weitere Unterordner befinden und Datenelemente hinzugefügt werden. Jeder Ordner erhält auch einen Bezeichner. Wird kein Bezeichner angegeben erhält der Nutzer einen Hinweis dies nachzuholen (siehe Abbildung 3.15).

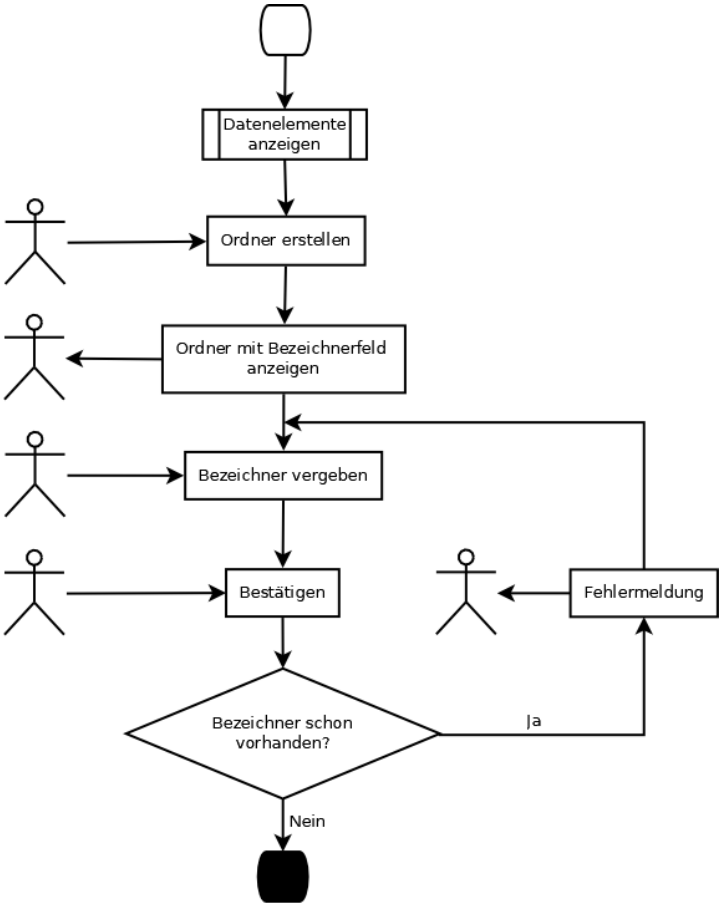


Abbildung 3.15: Flussdiagramm: Ordner erstellen

Datenelemente und Ordner löschen

Jedes Datenelement und Ordner eines Knotens kann über Betätigung eines Buttons auch wieder entfernt werden. Ein Hinweis soll versehentliches Löschen verhindern. Nach Bestätigung wird das Datenelement oder der Ordner entfernt, bei Abbruch wird nicht gelöscht. Wird ein Ordner gelöscht, werden auch alle darin enthaltenen Datenelemente und Ordner entfernt. Um versehentliches Löschen enthaltener Datenelemente und Ordnern zu verhindern, wird im zuvor erschienenen Hinweis auf vorhandene Datenelemente und Ordner hingewiesen (siehe Abbildung 3.16).

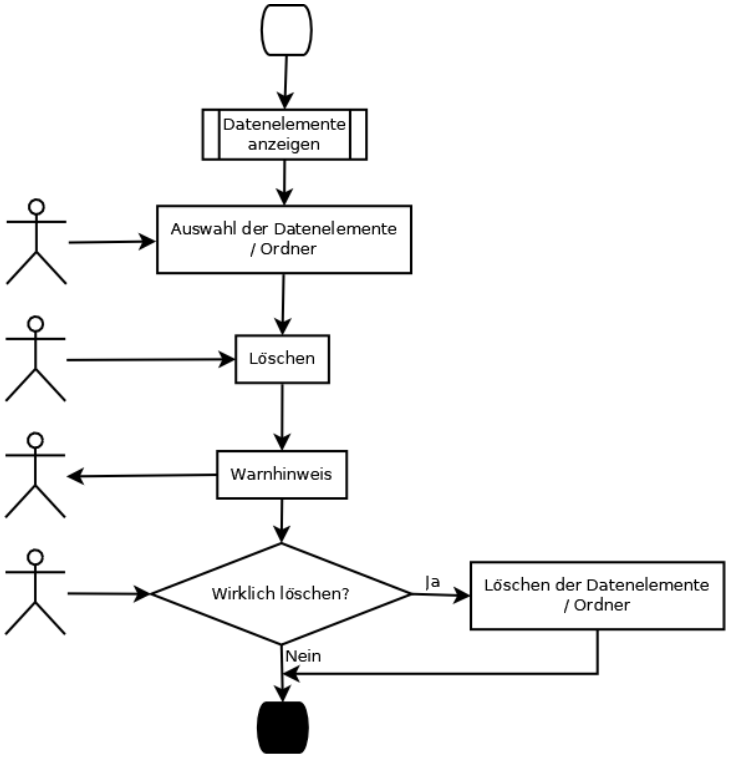


Abbildung 3.16: Flussdiagramm: Datenelemente und Ordner löschen

Datenelemente und Ordner umbenennen

Die Bezeichnung eines Datenelements oder eines Ordners kann durch den Benutzer angepasst werden (siehe Abbildung 3.17).

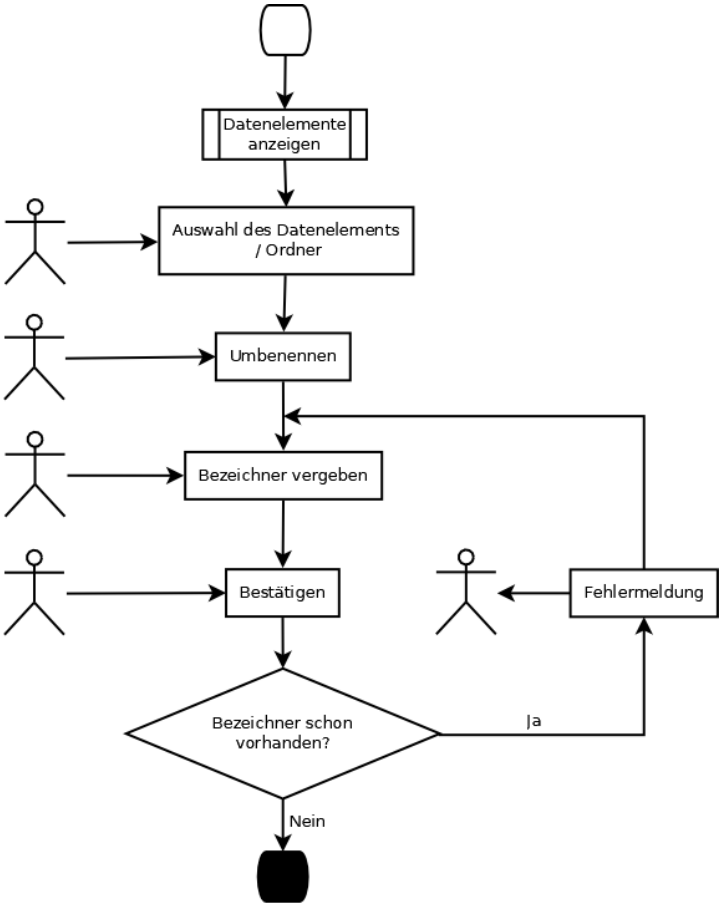


Abbildung 3.17: Flussdiagramm: Datenelemente und Ordner umbenennen

3.2.2 Spezifizierte Input-, Output- und Hybrid-Datenelemente

Wie Abbildung 3.18 zeigt, können jedem Task Template oder jeder Taskinstanz Datenelemente als Startvoraussetzung (Input-Daten), als Ergebnis (Output-Daten) oder beides (Hybrid-Daten) festgelegt werden. Für die Datenelemente können Platzhalter definiert werden. Diese Spezifikationen legen den Bezeichner und den Dateityp fest, den die zu einem späteren Zeitpunkt hinzugefügte unstrukturierte Datei haben soll. Unstrukturierte Daten die eine Spezifikation nicht erfüllen, können diese also nicht ersetzen. Schon bestehende Spezifikationen können geändert oder gelöscht werden. Bis auf zwei Ausnahmen können auf Datenelemente die als Input, Output oder Hybrid festgelegt sind die selben Operationen wie die unter Kapitel 3.2.1 beschriebenen durchgeführt werden. Einzig das Verlinken von Datenelementen und das Erstellen von Ordnern bei den Output Datenelementen ist hier nicht möglich.

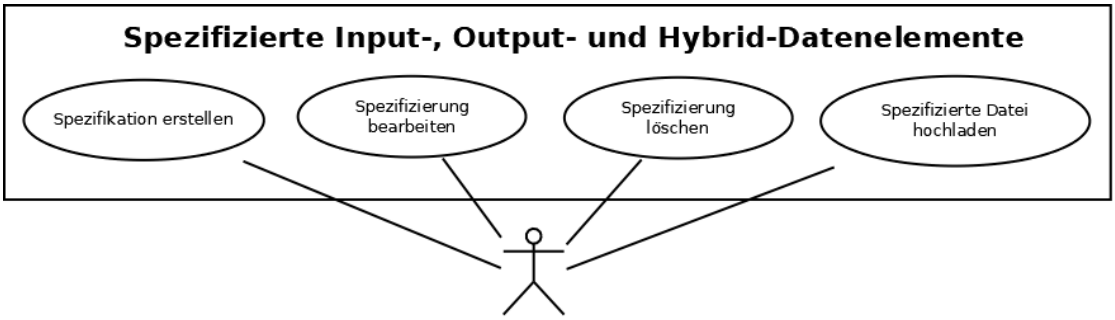


Abbildung 3.18: Spezifizierte Input-, Output- und Hybrid-Datenelemente

Spezifikation erstellen

Um die Spezifikation vornehmen zu können, muss zunächst festgelegt werden, ob es sich um einen Input-, Output- oder Hybrid-Datenelement handelt. Anschließend wird der Benutzer aufgefordert einen Dateinamen und einen Dateityp (z.B. .doc, .pdf, usw.) anzugeben. Nach Bestätigung der Angaben überprüft das System, ob die Kombination aus Bezeichner und Dateityp im vorliegenden Kontext einzigartig ist. Ist dies nicht der Fall wird der Benutzer aufgefordert eine Kombination aus Bezeichner und Dateityp zu wählen, die noch nicht vorhanden ist. Sind die Eingaben korrekt, wird die Spezifikation übernommen (siehe Abbildung 3.19).

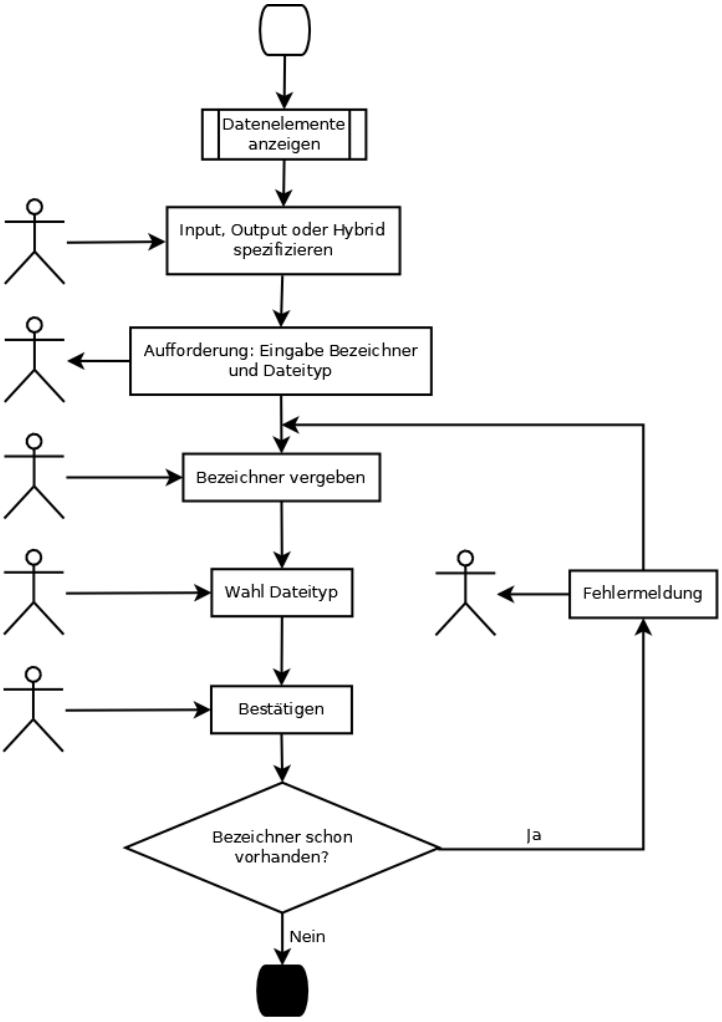


Abbildung 3.19: Flussdiagramm: Input, Output und Hybrid spezifizieren

Spezifizierung bearbeiten

Die Bearbeitung einer Spezifizierung erlaubt dem Benutzer den Bezeichner oder den Dateityp zu ändern. Auch nach einer Änderung findet eine Überprüfung der Kombination aus Bezeichner und Dateityp auf Einzigartigkeit statt. Erst wenn die Kombination im vorliegenden Kontext einzigartig ist, kann die Änderung übernommen werden (siehe Abbildung 3.20).

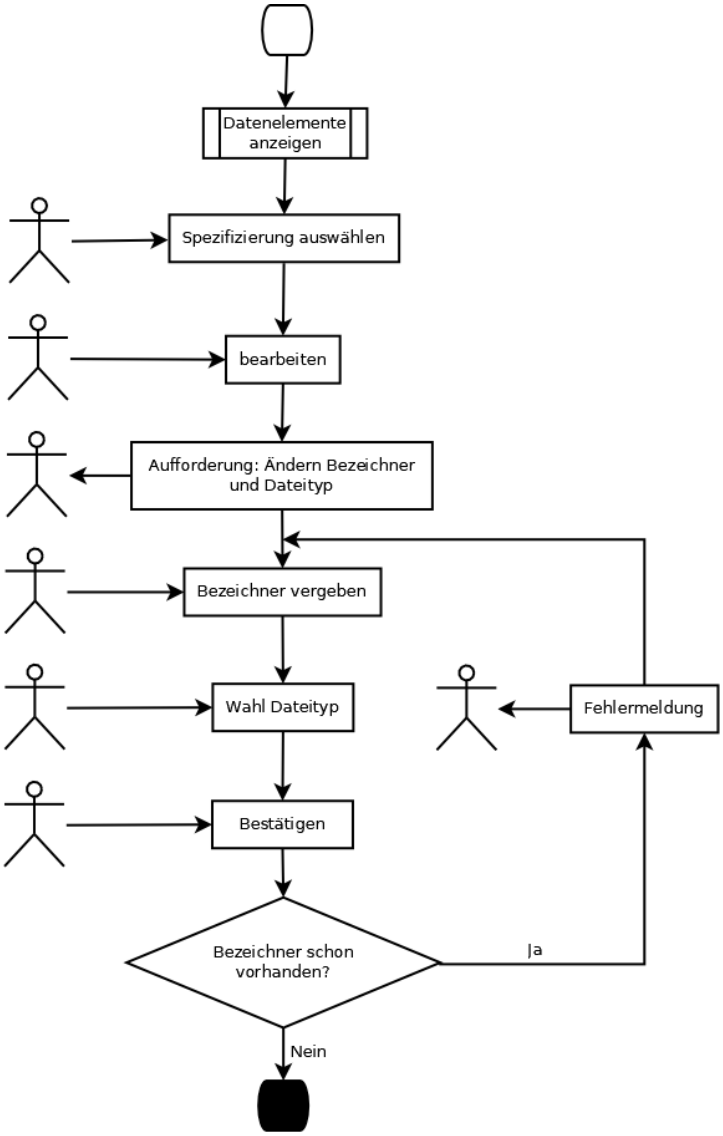


Abbildung 3.20: Flussdiagramm: Spezifizierung bearbeiten

Spezifizierung löschen

Das Löschen einer Spezifizierung entfernt diese aus dem jeweiligen Knoten. Auch in diesem Fall muss das Löschen durch den Benutzer vor Durchführung nochmals bestätigt werden (siehe Abbildung 3.21).

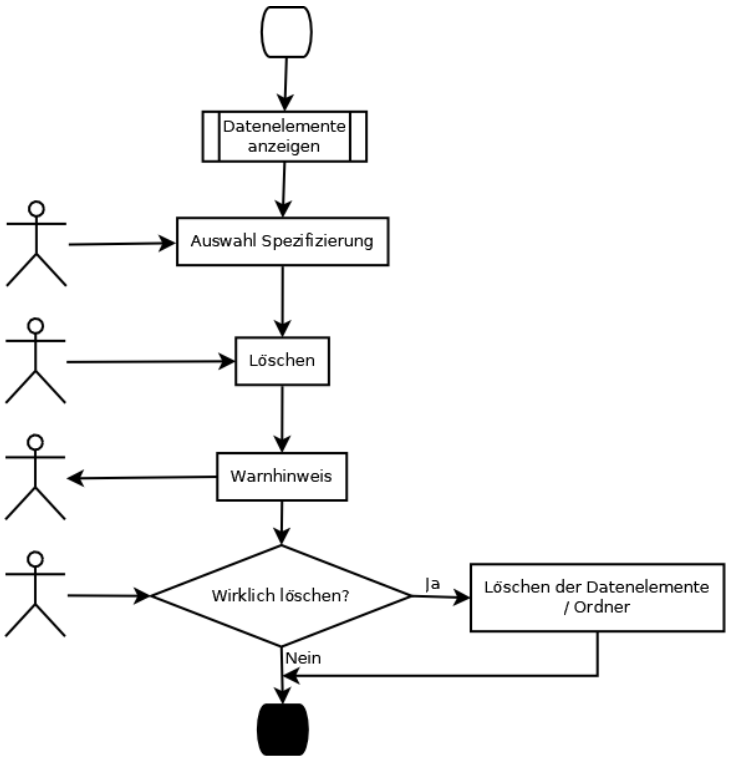


Abbildung 3.21: Flussdiagramm: Spezifizierung löschen

Spezifizierte Datei hochladen

Eine Spezifikation kann durch unstrukturierte Daten, die diese erfüllen, ersetzt werden. Der Nutzer wählt die zu ersetzende Spezifikation aus und wählt „Datei hochladen“. Über einen Dateiselektor kann die gewünschte Datei ausgewählt werden. Nach Betätigung von Hinzufügen wird die Datei gegen die Spezifikation geprüft. Stimmen Dateinamen und Dateityp mit den in der Spezifikation festgelegten Bezeichner und Dateityp überein, wird die Spezifikation entfernt und durch die hochgeladene unstrukturierte Datei ersetzt. Findet keine Übereinstimmung statt, kann eine andere Datei zum hochladen ausgewählt werden (siehe Abbildung 3.22).

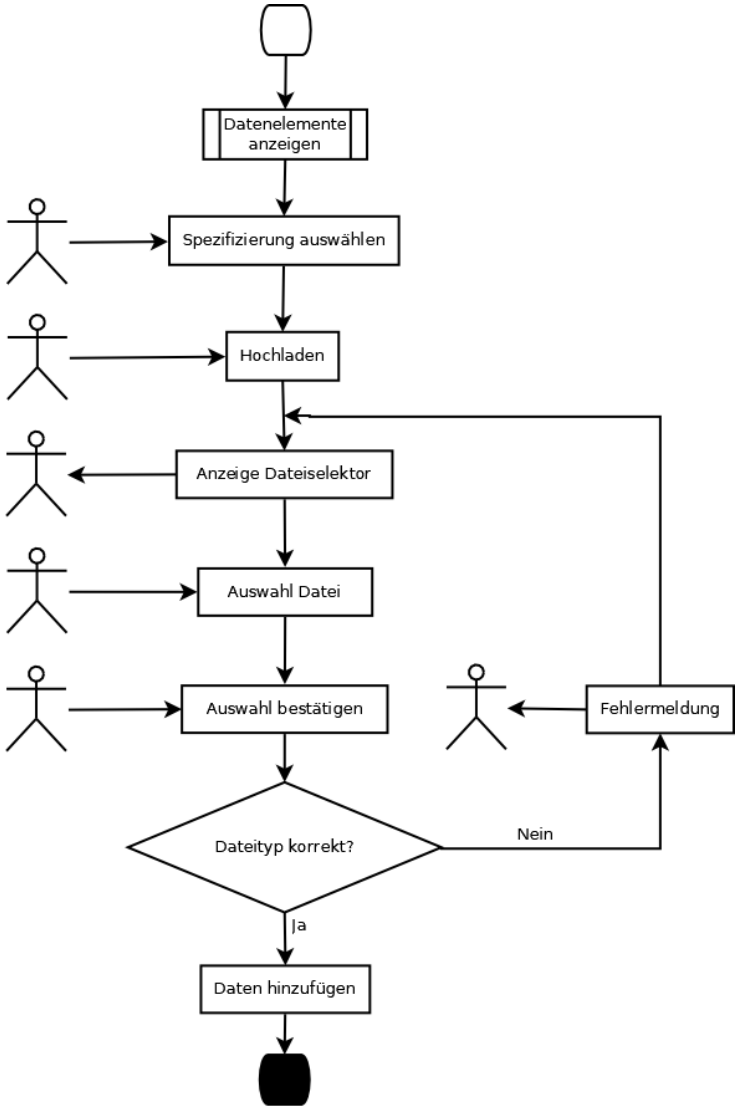


Abbildung 3.22: Flussdiagramm: Spezifizierte Datei hochladen

3.2.3 Datenübersicht in Prozessinstanz und Prozess Template

Abbildung 3.23 zeigt welche Anforderungen an die Datenübersicht in einer Prozessinstanz und einem Prozess Template gestellt werden. Für einen schnellen Überblick über Datenelemente, die in einem Prozess Template oder einer Prozessinstanz zuletzt hinzugefügt wurden, wird eine zentrale Übersicht benötigt. Eine Sortierfunktion nach dem Erstellungsdatum erhöht dabei die Übersicht. Weiter ist es es möglich, Datenelemente zu löschen und unstrukturierte Daten herunterzuladen und strukturierte Daten zu bearbeiten.

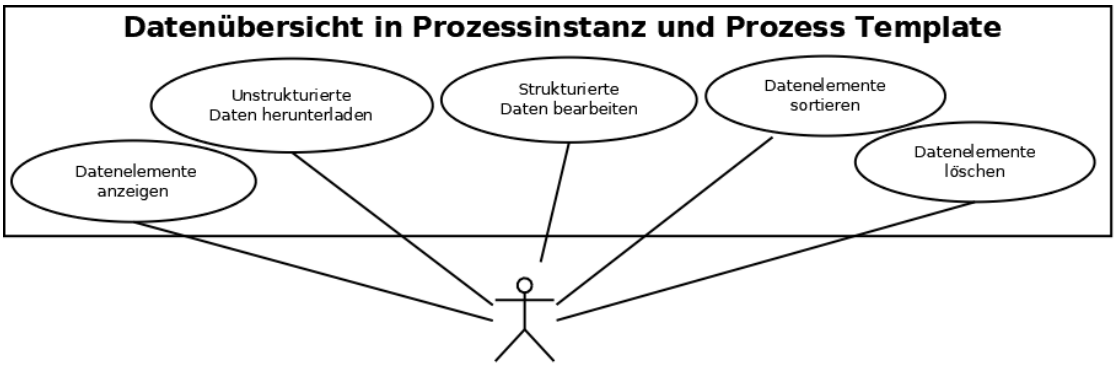


Abbildung 3.23: Use-Case Diagramm zur Datenübersicht in Prozessinstanz und Prozess Template

Unstrukturierte Daten herunterladen

Der Nutzer kann die vorhandenen Datenelemente direkt aus der Übersicht herunterladen. Im „Speichern unter“-Dialog des Betriebssystems, auf dem der Web-Client ausgeführt wird, kann der Benutzer den Speicherort selbst auswählen (siehe Abbildung 3.24).

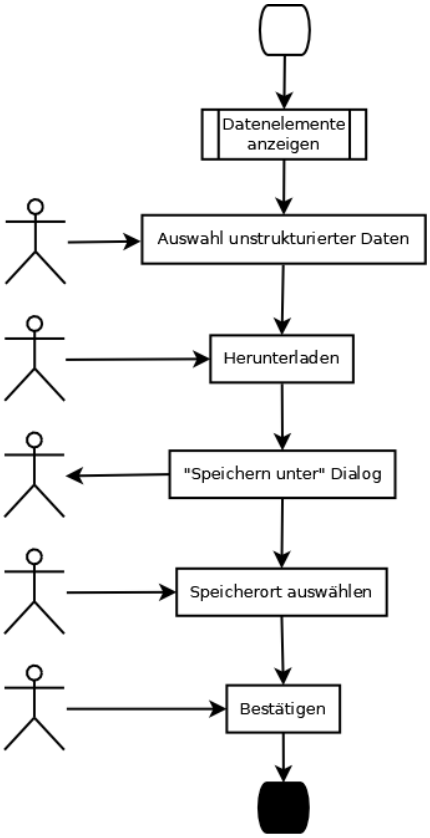


Abbildung 3.24: Flussdiagramm: Unstrukturierte Daten herunterladen

Strukturierte Datei bearbeiten

Über einen spezifischen Button können strukturierte Daten bearbeitet werden. Der Bezeichner der strukturierten Datei kann nicht geändert werden. Änderungen können nur an den Attributswerten vorgenommen werden, dabei ist auf den Datentyp zu achten. Entspricht die Eingabe des Nutzers nicht dem benötigten Datentyp, wird durch einen Hinweis darauf aufmerksam gemacht und auf den Dateityp hingewiesen (siehe Abbildung 3.25).

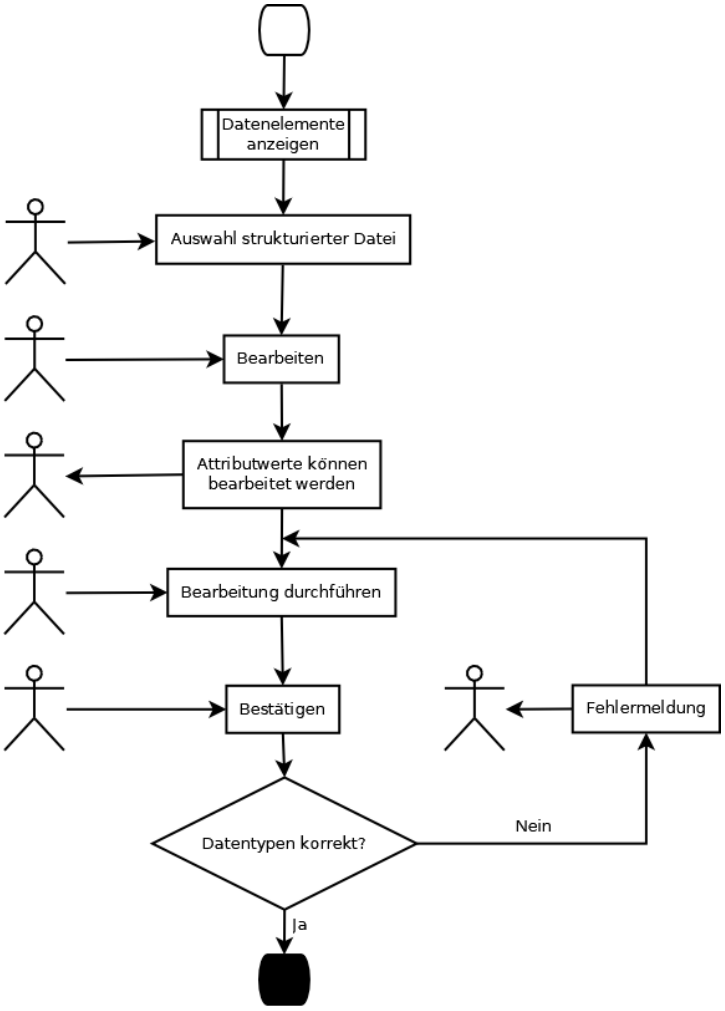


Abbildung 3.25: Flussdiagramm: Strukturierte Datei bearbeiten

Datenelement löschen

Jedes der zuletzt hinzugefügten Datenelemente kann hier durch die Betätigung eines Buttons wieder entfernt werden. Ein Hinweis soll in diesem Fall ein versehentliches Löschen verhindern. Nach Bestätigung wird das Datenelement entfernt. Bei Abbruch wird das ausgewählte Datenelement nicht gelöscht (siehe Abbildung 3.26).

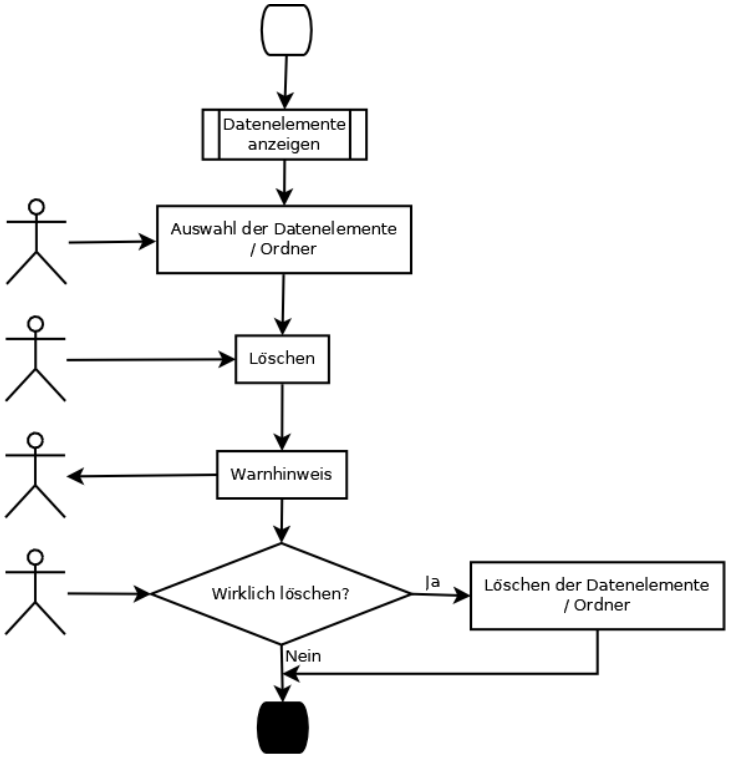


Abbildung 3.26: Flussdiagramm: Datenelement löschen

Datenelemente sortieren

Die Übersicht über die Datenelemente soll mit einer Sortierfunktion erweitert werden. Damit lassen sich Datenelemente nach gemeinsamen Eigenschaften gruppieren um dadurch die Übersichtlichkeit zu erhöhen (siehe Abbildung 3.27).

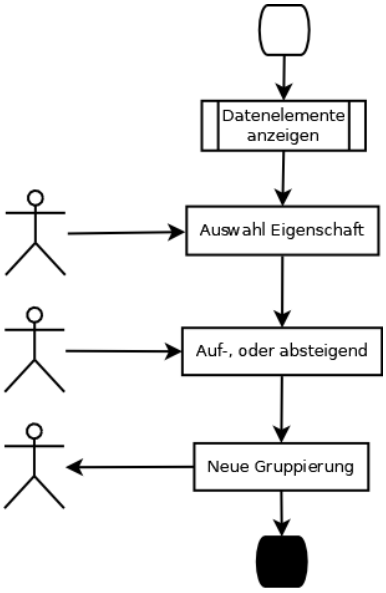


Abbildung 3.27: Flussdiagramm: Datenelemente sortieren

3.2.4 Verwaltung strukturierter Daten Templates

Dem Administrator unterliegt die Verwaltung der strukturierten Daten Templates, die in proCollab, an allen Orten die strukturierte Daten aufnehmen, als Vorlage verwendet werden können (siehe Abbildung 3.28). Dies ermöglicht sowohl das Betrachten vorhandener Templates, als auch das Erstellen von beliebig vielen neuen strukturierten Daten Templates. Darüber hinaus ist es möglich bestehende strukturierte Daten Templates zu ändern und zu löschen.

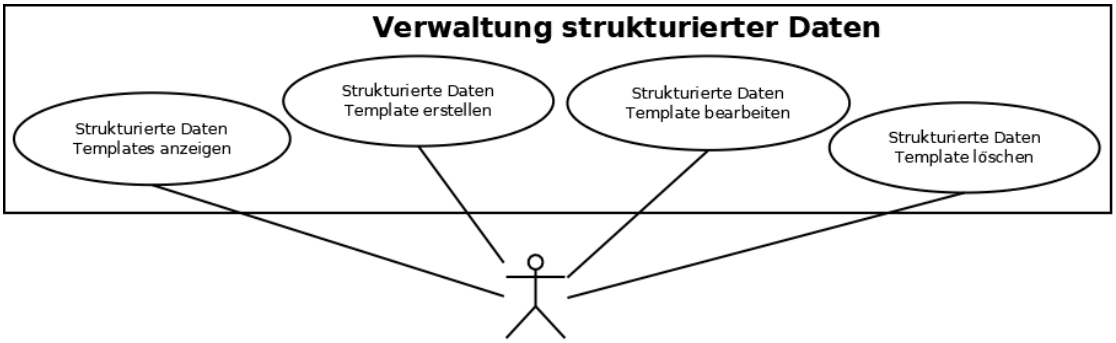


Abbildung 3.28: Use-Case Diagramm zur Verwaltung strukturierter Daten

Strukturierte Daten Templates anzeigen

In einer Übersicht werden alle von einer berechtigten Rolle erstellten strukturierten Daten Templates angezeigt. Darüber hinaus können dort neue strukturierte Daten Templates erstellt, geändert oder gelöscht werden (siehe Abbildung 3.29).

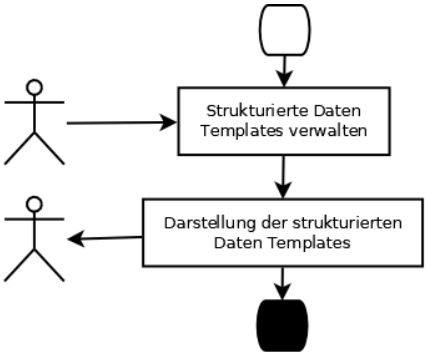


Abbildung 3.29: Flussdiagramm: Strukturierte Daten Templates anzeigen

Strukturierte Daten Templates erstellen

Bevor strukturierte Daten innerhalb eines Prozess Templates oder einer Prozessinstanz verwendet werden können, müssen diese von einem Administrator erstellt werden. Eine unstrukturierte Datei benötigt eine Bezeichnung, die diese möglichst genau beschreibt. Anschließend können beliebig viele Attribute hinzugefügt werden, bestehend aus einer Bezeichnung und einem Datentyp (siehe Abbildung 3.30).

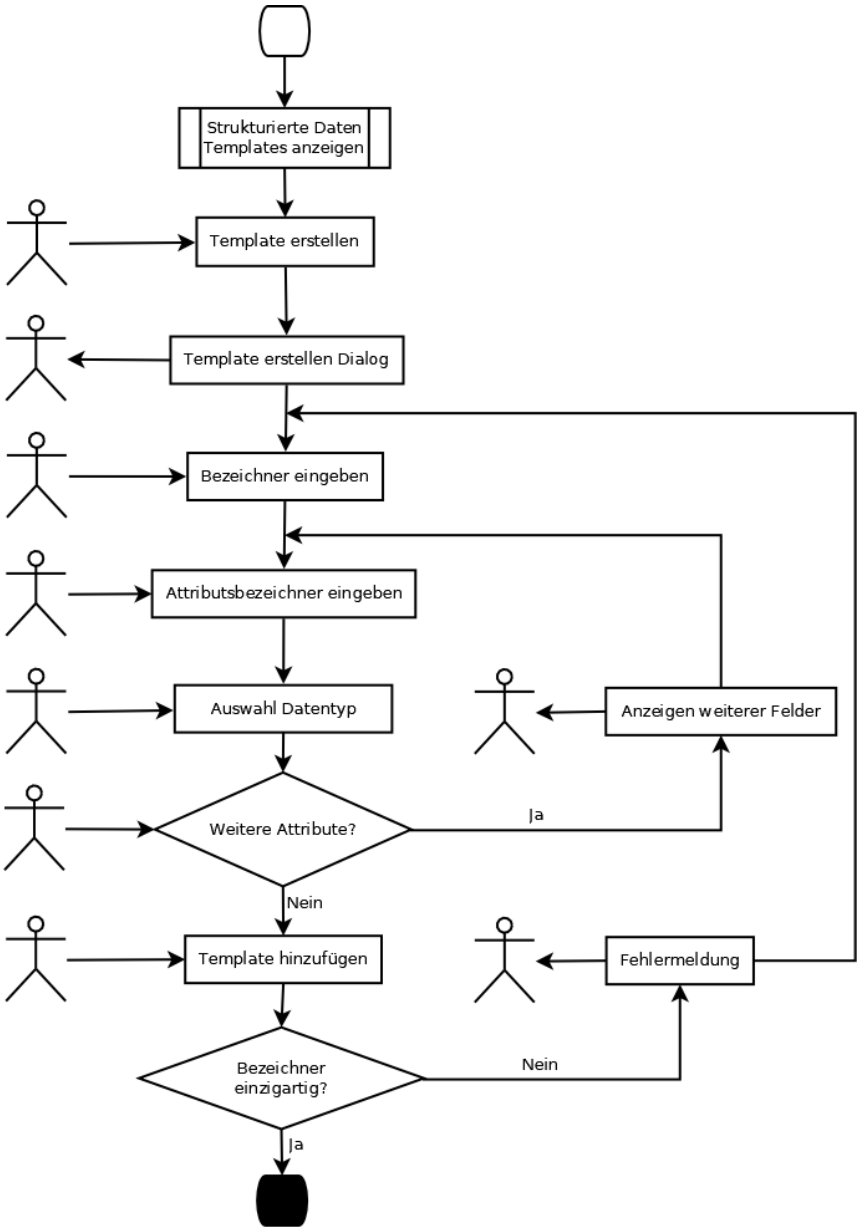


Abbildung 3.30: Flussdiagramm: Strukturierte Daten Templates erstellen

Strukturierte Daten Templates ändern

Eine existierende strukturierte Datei kann geändert werden, solange sie nicht schon innerhalb eines Prozess Templates oder einer Prozessinstanz verwendet wird. Der Benutzer erhält bei der Änderung schon verwendeter strukturierter Daten einen Hinweis auf diesen Konflikt (siehe Abbildung 3.31).

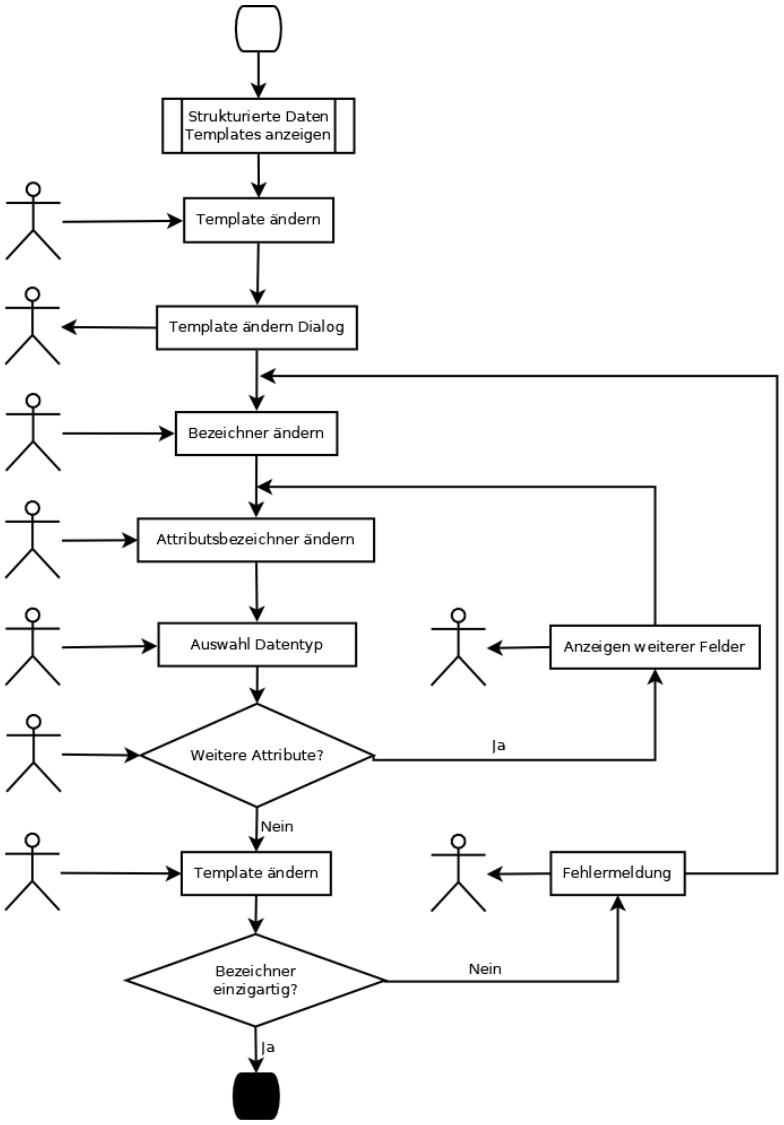


Abbildung 3.31: Flussdiagramm: Strukturierte Daten Templates ändern

Strukturierte Daten Templates löschen

Daten Templates lassen sich wieder löschen. Wurde ein Template für strukturierte Daten schon innerhalb eines Prozess Templates oder einer Prozessinstanz genutzt, werden diese ebenfalls an ihren jeweiligen Orten entfernt. Auch hier muss der Benutzer auf diesen Schritt hingewiesen werden, um versehentliches Löschen vorzubeugen (siehe Abbildung 3.32).

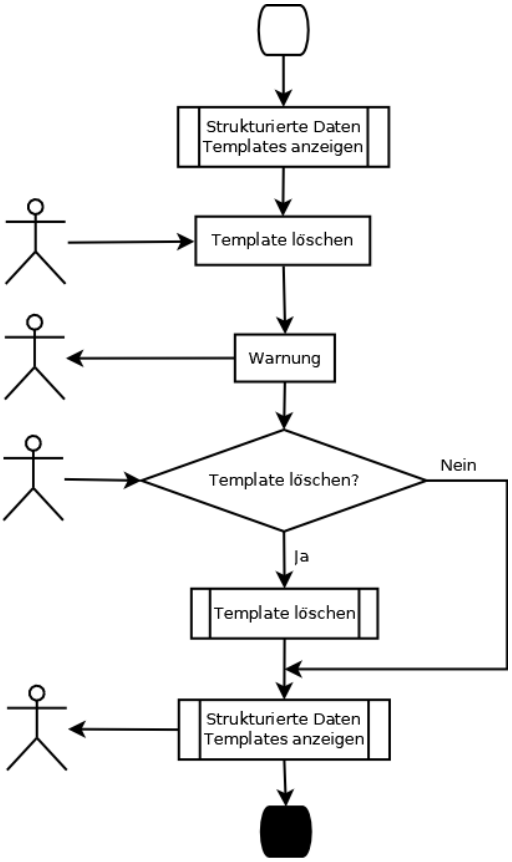


Abbildung 3.32: Flussdiagramm: Strukturierte Daten Templates löschen

3.3 Entwicklungskontext

Für das folgende Kapitel und vor allem auch für die spätere Implementierung (siehe Kapitel 5) ist es wichtig zu wissen, für welches Zielsystem das User Interface entwickelt

werden soll. Denn die verwendete Hard- und Software hat Einfluss auf die Umsetzungsmöglichkeiten der Benutzeroberfläche.

3.3.1 Hardware

Bei proCollab handelt es sich um eine Webanwendung, deren Programmlogik und Datenbank auf einem Webserver läuft und die Benutzerschnittstelle im Webbrowser des Anwenders ausgeführt wird. Für die Entwicklung eines UI-Konzepts ist vor allem die Hardware, auf der der Client läuft, interessant. Bei einer Webanwendung ist die Hardwarebasis sehr vielseitig und kann von einem Desktop-PC, über Laptops, bis hin zum Smartphone der unterschiedlichsten Leistungsklassen reichen. Das Gerät auf dem der Client laufen soll, muss also zumindest die Mindestanforderungen des benutzten Webbrowsers erfüllen. Für die Entwicklung einer Benutzeroberfläche ist neben den reinen Leistungsdaten des Endgeräts, vor allem das Display von besonderem Interesse. Diese können sich, durch ihre unterschiedlichen Größen und der dargestellten Auflösung stark auf die Entwicklung einer Benutzerschnittstelle auswirken.

Momentan beschränkt sich das proCollab Projekt auf die üblichen Größen von Displays moderner Desktops und Laptops. Für diese Arbeit wird deshalb, von den laut der angeführten Statistik, am häufigsten verwendeten Auflösungen und Display-Größen im Desktop- und Laptopbereich ausgegangen. Aus Tabelle 3.2 geht deutlich hervor, dass Auflösungen unter 1366 x 768 Pixel kaum noch eine Rolle spielen. Wenn man die Werte von W3Schools betrachtet [W3S18a], fällt auf, dass sogar 50 % der Nutzer eine Auflösung von 1920 x 1080 Pixel verwenden. Dementsprechend soll der Entwurf auf diese Auflösungen ausgelegt sein.

Quelle	höher	1920 x 1080	1366 x 768	1280 x 1024	1280 x 800
StatCounter	—	17,95 %	28,75 %	4,44 %	4,90 %
W3Schools	32.90 %	18,00 %	34,00 %	4,00 %	3,00 %

Tabelle 3.2: Meistgenutzte Desktopauflösung Januar 2018 [Sta18b, W3S18a]

3.3.2 Software

proCollab ist eine Webanwendung, die einen aktuellen Webbrowser voraussetzt. Dadurch wird die Gestaltung der Benutzeroberfläche betriebssystemunabhängig und ist

somit nicht an den Styleguide des Betriebssystems (z.B. Form und Anordnung von Buttons) gebunden. Ebenso ist es durch die Verwendung eines Webbrowsers als Client einfacher, ein einheitliches und vom Betriebssystem unabhängiges „Look and Feel“ der Benutzeroberfläche umzusetzen.

Laut den Statistiken von StatCounter [Sta18a] und W3Schools [W3S18b] (siehe Tabelle 3.3) ist Google Chrome mit Abstand der am meisten verbreitete Webbrowser im Desktopbereich. Auf Platz zwei folgt ebenfalls mit noch deutlichem Abstand zu den nachfolgenden Browsern der Firefox von Mozilla. Demnach ist sicherzustellen, dass die Implementierungen in diesen beiden Webbrowsern ohne Fehler funktionieren.

Quelle	Chrome	Firefox	Internet Explorer	Safari	Edge	Opera
StatCounter	65,98 %	11,87 %	7,28 %	5,87 %	4,11 %	—
W3Schools	77,20 %	12,40 %	incl. Edge 4,10 %	3,20 %	—	1,60 %

Tabelle 3.3: Marktanteile von Desktop Webbrowser im Januar 2018 [Sta18a, W3S18b]

4

Konzept

Die in Kapitel 3 definierten Anforderungen werden nun in konkreten Benutzeroberflächen umgesetzt. Das Ziel ist es eine aufgabenangemessene Benutzerschnittstelle zu entwerfen. Dabei sollen aber auch die in Kapitel 2 beschriebenen Gestaltungsrichtlinien Berücksichtigung finden.

4.1 Konzeptuelles User-Interface-Modell

Das Konzeptuelle User-Interface-Modell bildet die Grundlage für den Mockup-Entwurf [May10]. Es werden Darstellungsregeln erarbeitet, die als Fundament für die weitere Dialoggestaltung dienen sollen. Die Dialogsstruktur hilft herauszufinden wo die Dialoge, für die in Kapitel 3.2 definierten Anforderungen, untergebracht werden sollen.

4.1.1 Darstellungsregeln aus dem vorhandenen Client

Um ganzheitlich eine konsistente und erwartungskonforme Benutzeroberfläche zu erhalten müssen die Gestaltungskonventionen des Clients festgehalten werden. Es gilt herauszufinden, wie einzelne Dialogelemente bisher im Client gestaltet, angeordnet und verwendet werden. Basierend auf diesen übergeordneten Darstellungsregeln und den daraus entstehenden Einschränkungen entsteht in Kapitel 4.2 der gesamte User-Interface Entwurf.

Inhaltsbereich

Die große freie Fläche in Abbildung 4.1 ist der Inhaltsbereich und dient der Darstellung der Inhalte des ausgewählten Menüpunktes. Am oberen Ende befindet sich eine graue

Menüzeile, diese kann grüne Buttons aufnehmen die linksbündig angeordnet sind. Am rechten Ende der Menüzeile befindet sich die Bezeichnung des aktuellen Inhalts des Inhaltsbereichs. Darunter folgt nun der eigentliche Inhalt, der recht frei gestaltet werden kann. Alle nachfolgend angeführten Elemente und natürlich neu entwickelte können hier dargestellt werden.

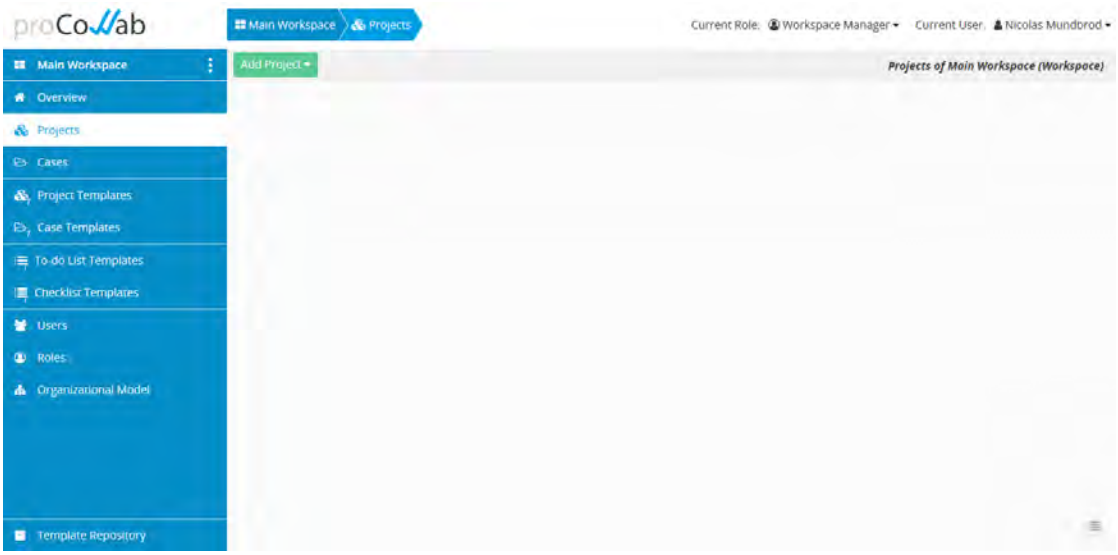


Abbildung 4.1: Inhaltsbereich

Modale Dialoge und ihre Elemente

Ein Dialog wird modal genannt, wenn Benutzer mit ihm interagieren müssen, bevor sie zu den darunter liegenden Dialogen zurück gelangen können. Solange also ein modaler Dialog geöffnet ist, kann mit den darunter liegenden Dialogen nicht interagiert werden. Im Web-Client werden modale Dialoge ausschließlich innerhalb des Inhaltsbereichs, zwischen Menüleiste und Kopfzeile, angezeigt (siehe Abbildung 4.2). Im Kopf befinden sich der Titel des Dialogs und ein Button zum Schließen des Dialogs am rechten Rand. In der Kopfzeile dürfen sich sonst keine weiteren Buttons befinden. Direkt darunter beginnt der Inhaltsbereich dieser kann als oberstes Steuerelement eine Schaltfläche enthalten, die es ermöglicht zwischen mehreren Registerkarten zu wechseln. Danach können weitere Elemente wie Eingabefelder oder Dropdown-Felder folgen, die die gesamte Dialogbreite einnehmen. Der Bezeichner der Elemente befindet sich immer darüber. Buttons zum durchführen der Aktion oder dem Beenden des Dialogs befinden sich in

der Ecke rechts unten. Dabei sind Buttons zur Durchführung bestimmter Aktionen grün, Bearbeiten gelb, Löschen rot und Abbrechen weiß.

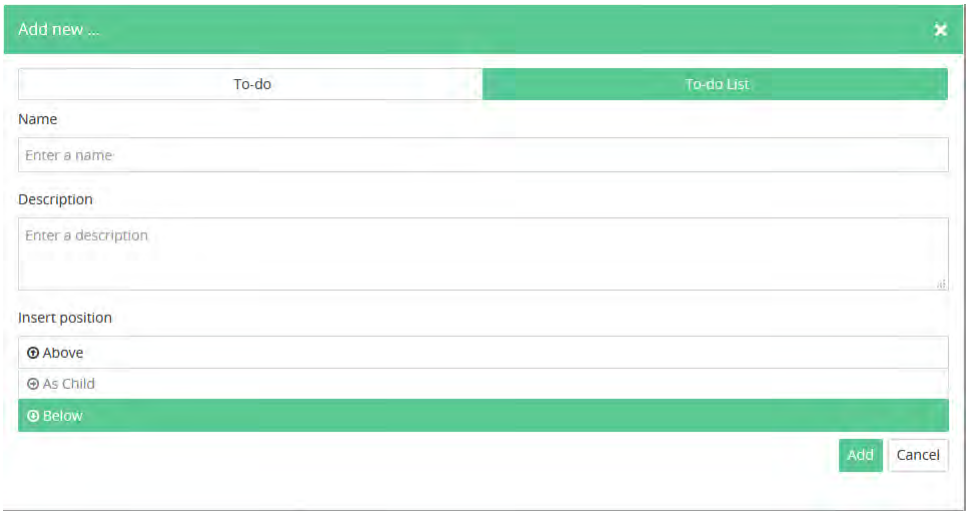


Abbildung 4.2: Beispiel eines modalen Dialogs

Container

Der proCollab Web-Client nutzt Container um Informationen gut zugänglich und kompakt unterzubringen (siehe Abbildung 4.3). So werden diese unter anderem genutzt um Prozessinstanzen und Prozess Templates im ungeöffneten Zustand übersichtlich darzustellen. Im Kopf des Containers befindet sich der Containertyp-Bezeichner. Im Inhaltsbereich, an oberster Stelle, steht der genau Bezeichner des Inhalts. Darunter können allgemeine Informationen über den Inhalt folgen. Am unteren rechten Rand des Containers befinden sich Buttons um mit dem Containerinhalt interagieren zu können.

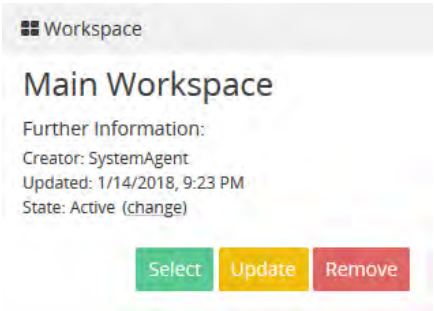


Abbildung 4.3: Beispiel eines Containers

4.1.2 Dialogstruktur

Aus der in Kapitel 3.3 durchgeführten Aufgabenanalyse lassen sich verschiedene Ansichten herleiten und zu einer Dialogstruktur (siehe Abbildung 4.4) zusammenfügen. Das Ziel dieser Dialogstruktur ist es, die Hauptnavigationswege zwischen den verschiedenen Ansichten festzulegen und gleiche oder sich stark ähnelnde Ansichten aufzufinden. Das Identifizieren gleicher oder ähnlicher Anschichten vereinfacht die Erstellung der User-Interface Mockups, so können Entwürfe für gleiche oder ähnliche Ansichten wiederverwendet werden. Dies stellt zusätzlich sicher, dass die Dialoge über alle Anwendungsfälle hinweg konsistent gestaltet werden.

Dabei werden die Dialoge in Rechtecken dargestellt, während die Übergänge zwischen ihnen durch Pfade symbolisiert werden. Außerdem ist bei den Dialogen gekennzeichnet, ob sie modal oder nicht modal sind, wobei nur bei den modalen Dialogen eine Anmerkung steht. Jeder nicht als modal gekennzeichnete Dialog, ist also automatisch nicht modal.

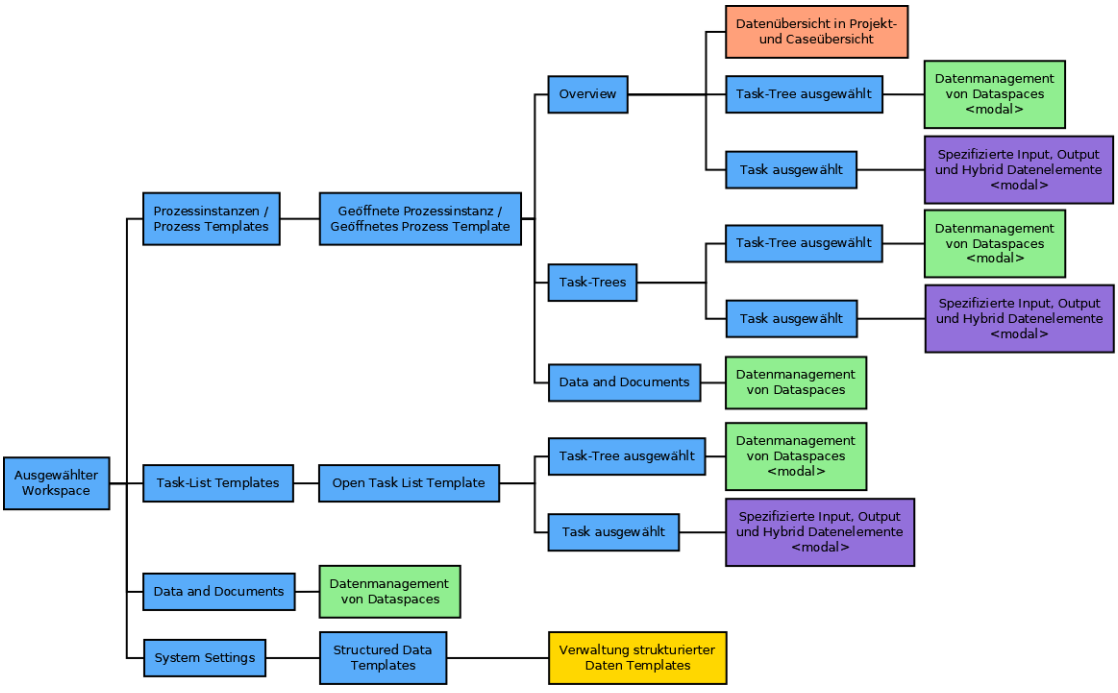


Abbildung 4.4: Dialogstruktur

4.2 User-Interface Mockups

Aus den in der Aufgabenanalyse definierten Vorgaben werden nun, unter Berücksichtigung der Richtlinien zur Dialoggestaltung und den gerade betrachteten Darstellungsregeln, grafische UI-Mockups entwickelt. Das Ziel ist es eine möglichst einheitliche und erwartungskonforme Darstellung sowohl der neuen Dialoge untereinander, als auch zu den im Web-Client vorhandenen Dialogen zu schaffen. Für einige Anforderungen wurden mehrere Lösungsansätze entwickelt, die nun diskutiert werden sollen um anschließend die Entscheidung für einen Ansatz begründen zu können.

4.2.1 Grundsätzliche Darstellung von Datenelementen

Die wohl bekannteste Darstellung einer großen Anzahl an Datenelementen ist, ähnlich wie in Kapitel 3.1.2 in Abbildung 3.5 bei Dropbox zu sehen, die Liste. Alternativ dazu können Datenelemente auch in einer Baumstruktur dargestellt werden. Die Verwendung einer Liste oder eines Baums beeinflusst wie durch die Datenelemente navigiert wird und diese dargestellt werden. Durch eine Liste wird in die Tiefe navigiert, in dem Ordner durch anklicken geöffnet werden. Höhere Ebenen werden durch eine Brotkrümelnavigation (siehe Abbildung 4.5), die den Pfad zum aktuell geöffneten Ordner anzeigt, erreicht. Eine Liste beschränkt sich ausschließlich auf die Darstellung der Datenelemente innerhalb des gerade geöffneten Ordners. Datenelemente die außerhalb oder in weiteren Unterordnern des gerade geöffneten Ordners liegen, werden nicht angezeigt.

Abbildung 4.5: Entwurf einer Liste zur Darstellung von Datenelementen

Ein Baum stellt Datenelemente und Ordner als hierarchische Liste dar, wobei die Ordner als Knoten und die Datenelemente als Blätter des Baumes dienen. In die Tiefe des Baumes wird durch Öffnen der Knoten navigiert. Die darin enthaltenen Knoten und

Datenelemente werden daraufhin etwas eingerückt angezeigt. Im Gegensatz zu einer Liste zeigt die Baumansicht immer alle geöffnete Knoten und die darin enthaltenen geschlossenen Knoten und Datenelemente. Es ist somit möglich die gesamte Verzeichnisstruktur mit ihren Datenelemente auf einmal zu betrachten. Um die Übersicht zu verbessern, lassen sich geöffnete Knoten wieder schließen, jedoch können Bäume trozdem unübersichtlich erscheinen.

Gegenüberstellung

Bei einem Vergleich der Vor- und Nachteile dieser beiden Darstellungsarten fällt auf, dass bei einer Liste nur der Inhalt des gerade geöffneten Ordners angezeigt wird. Es wird so auf das wesentliche konzetriert, wodurch die Übersicht erhöht werden kann, solange der Inhalt dem Benutzer trotzdem alle benötigten Informationen bereitstellt. Diese Übersichtlichkeit bleibt unabhängig der Komplexität der Verzeichnisstruktur immer gleich. Allerdings fehlt dadurch der Überblick über die gesamte Verzeichnisstruktur und das Navigieren zu Ordnern außerhalb des gerade geöffneten Pfades ist aufwändig.

In einer Baumdarstellung hingegen ist eine Übersicht über die gesamte Verzeichnisstruktur gegeben, wodurch sich schneller an den gewünschten Ort navigieren lässt. Ein großer Vorteil ist, dass in einem Baum mehrere Knoten zur selben Zeit geöffnet sein können, so lassen sich mehrere Verzeichnisse gleichzeitig betrachten was einige Klicks einsparen kann. Außerdem können dadurch Drag & Drop Operationen effizienter durchgeführt werden. Sind allerdings viele Knoten geöffenet wächst der Baum schnell in die Breite und Tiefe, wodurch die Übersichtlichkeit wiederum leiden kann.

Entwurf

Die Baumstruktur wird wegen ihrer höherer Gesamtübersicht, der Möglichkeit mehrere Verzeichnisse gleichzeitig zu betrachten und der besseren Unterstützung für Verschiebevorgänge (Drag & Drop) gegenüber der Listenansicht gewählt. Abbildung 4.6 zeigt einen Entwurf einer Baumdarstellung, wie er für die weiteren Mockups verwendet werden soll. Durch das Hinzufügen von zusätzlichen Spalten zu dem Baum, können neben dem Namen noch weitere Informationen, wie der Name des Erstellers, Änderungsdatum und die Dateigröße dargestellt werden. Um die Individualisierbarkeit zu erhöhen, kann der Benutzer die Datenelemente nach jeder Spalte auf- oder absteigend sortieren indem er in der Kopfzeile auf die entsprechende Spalte klickt. In der rechten Spalte ist, mit Klick auf die drei vertikal angeordneten Punkten, zu jedem Datenelement und Ordner ein Dropdown-Menü aufrufbar. Darüber lassen sich viele der in der Aufgabenanalyse

definierten Anforderungen aufrufen. So können von dort unter anderem die Operationen wie das Löschen, Herunterladen, Verschieben, Kopieren und Umbenennen von Datenelementen eingeleitet werden.

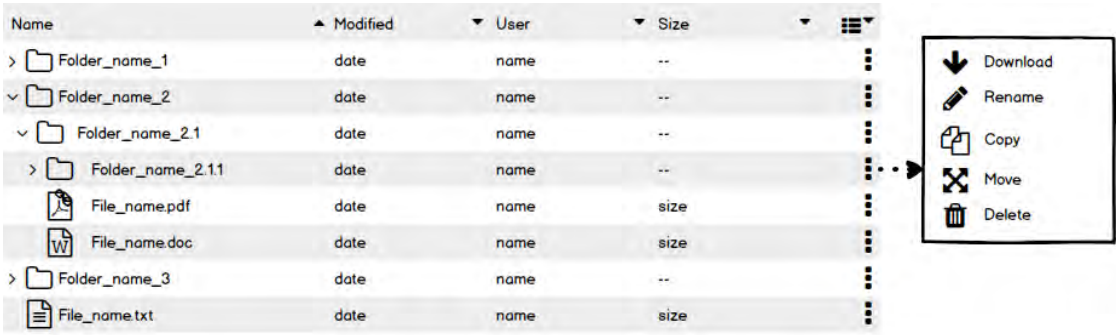


Abbildung 4.6: Entwurf: Baumdarstellung der Datenelemente

Thumbnail-Ansicht

Eine beliebte alternative Darstellungsart für Datenelemente, die auch von allen in Kapitel 3.1.2 vorgestellten vergleichbaren Systemen genutzt wird, ist die Thumbnail-Ansicht. Für den Benutzer kann die Darstellung der Datenelemente als Vorschaubild bei der Suche nach dem gesuchten Datenelement von Vorteil sein. Das schnelle und korrekte Erkennen des gesuchten Datenelementes durch den Benutzer spart Zeit, denn das Vorschaubild reicht häufig bereits aus, um das richtige Datenelement zu identifizieren. Aus diesem Grund und um die Individualisierbarkeit für den Nutzter zu erhöhen, wird die Thumbnail-Ansicht in einem Raster als zusätzliche Darstellungsart neben der Baumstruktur umgesetzt.

Wie in Abbildung 4.7 zu sehen ist, zeigt das Thumbnail nicht nur ein Vorschaubild seines Inhalts, sondern enthält auch die Angaben zu Dateinamen, Dateityp, Namen des Erstellers und des Erstellungsdatums. Über die drei vertikal angeordneten Punkte kann auch hier, wie schon bei der Baumdarstellung, das Dropdown-Menü für zusätzliche Operationen aufgerufen werden.

4.2.2 Datenmanagement von Data Spaces

Die Dialogstruktur in Kapitel 4.1.2 hat gezeigt, dass alle Data Spaces in proCollab dem Nutzer die selben Informationen und Operationen bereitstellen müssen. Die Heraus-

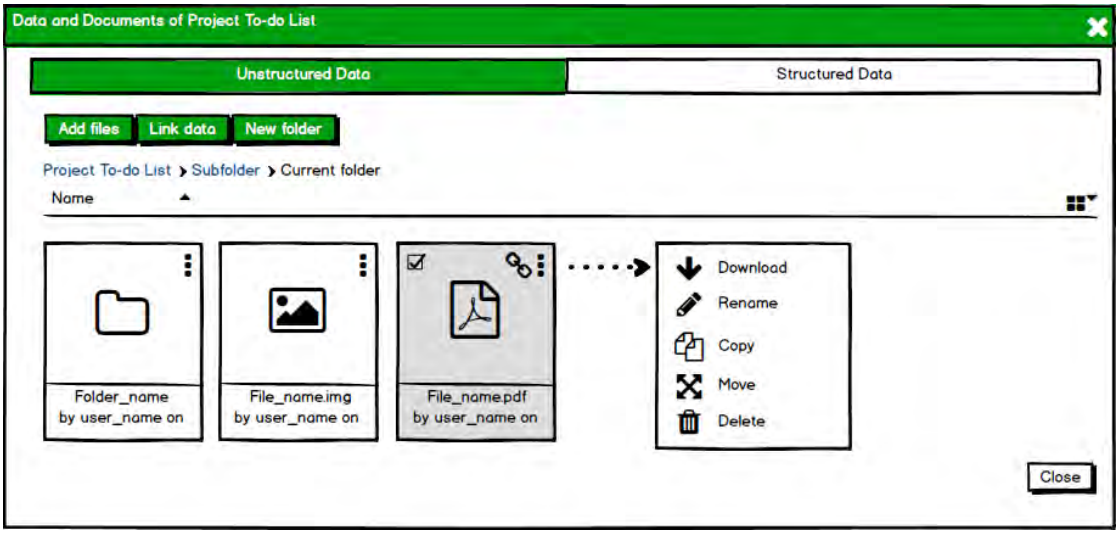


Abbildung 4.7: Thumbnailansicht von unstrukturierten Daten

forderung besteht darin die in Kapitel 4.2.1 entwickelte Baumdarstellung der Datenelemente so in den verschiedenen Ansichten unterzubringen, dass sie eine konsistente und erwartungskonforme Nutzererfahrung bereitstellen. Für den Mockup-Entwurf können die Ansichten nochmals aufgeteilt werden in die Task Tree- und der „Data and Documents“-Ansicht für den Workspace und die Prozesse. Während bei den „Data and Documents“-Ansichten komplett leere Inhaltsansichten zur Gestaltung und Unterbringung der Baumstruktur viel Platz bieten, ist dies bei den Task Tree-Ansichten nicht mehr der Fall. Dort können sich mehrere Listen und deren dazugehörige Tasks befinden. Hier muss die Baumdarstellung der Datenelemente in die bestehenden Elemente integriert werden, wodurch der Gestaltungsfreiraum stark einschränkt wird.

Task Tree-Ansicht

Es werden im Folgendem zwei komplett unterschiedliche Herangehensweisen gegenübergestellt, die jeweils zeigen wie eine Baumdarstellung der Datenelemente in diese Ansicht integriert werden kann.
In einem ersten Entwurf (siehe Abbildung 4.8) werden die Datenelemente in der Liste selbst angezeigt. Durch einen Klick auf die Liste erweitert sie sich, ähnlich eines Dropdown-Menüs, nach unten und die Datenelemente kommen zum Vorschein. Abbildung 4.9 zeigt als alternativen Entwurf, einen modalen Dialog, der sich über der Ansicht öffnet und die Funktionen zum Datenmanagement bereitstellt.

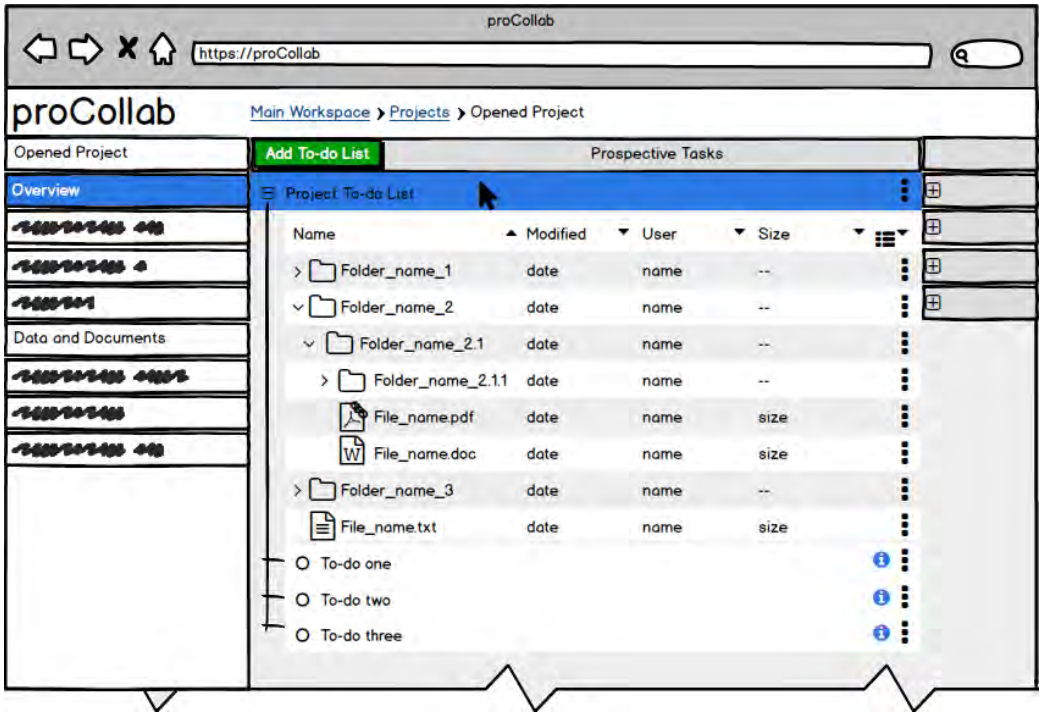


Abbildung 4.8: Projektübersicht mit expandierter To-do List

Gegenüberstellung

Der Dropdown Entwurf hat den Vorteil, dass er sich nahtlos in das vorhandene Design integriert und einen schnellen Überblick über die Datenelemente des Task Trees bietet. Allerdings wird auch ein Nachteil schnell offensichtlich, denn durch die Unterbringung der Baumstruktur direkt in der Liste, ist der Platz den der Baum in der Breite einnehmen kann sehr begrenzt. Dieser Platzmangel kann bei kleinen Browserfenstern, geringer Auflösung oder bei einem großen Baum mit vielen ausgeklappten Knoten, noch verstärkt werden. Generell ist die Spaltenbreite des Baumes sehr gering und für lange Dateinamen könnte es somit schnell zu eng werden. Die Darstellung der Datenelemente als Thumbnail ist durch diesen Platzmangel nicht sinnvoll. Ein weiterer Nachteil ist, dass die Ansicht in Prozessen mit vielen Task Trees und Tasks allein schon sehr voll sein kann und mit der zusätzlichen Darstellung der Datenelemente die Übersicht weiter negativ beeinflusst wird.

Das modale Pop-up Fenster zur Darstellung der Datenelemente beseitigt diese Nachteile. Es bietet ausreichend Platz in der Breite und Tiefe für eine übersichtliche Gestaltung der Baumdarstellung und Thumbnail-Ansicht der Datenelemente. Allerdings verliert

dieser modale Dialog gegenüber dem zuvor vorgestelltem Entwurf an einer schnellen Übersicht über die Datenelemente der Task Trees, da dieser die darunter liegende Ansicht verdeckt und erst geschlossen werden muss um erneut mit der ursprünglichen Ansicht interagieren zu können. Der größte Vorteil und Hauptgrund, weshalb der modale Dialog umgesetzt wird ist, dass er gegenüber dem ersten Entwurf, wie bei der „Data and Documents“-Ansicht eine leere Fläche mit ausreichend Platz zur Gestaltung des Dialogs zur Verfügung stellt. Somit ist das Ziel eine konsistente und erwartungskonforme Nutzererfahrung zu gewährleisten wesentlich einfacher zu erreichen.

Entwurf

Abbildung 4.9 zeigt nun den finalen Entwurf des Dialogs, der sich an die in Kapitel 4.1.1 festgelegten Konventionen von modalen Dialogen in proCollab hält. Ganz oben befinden sich die Tabs zum Umschalten zwischen unstrukturierten und strukturierten Daten. Darunter sind Buttons zum Hinzufügen und Verlinken von Datenelementen und dem Erstellen von Ordnern angebracht. Weiter darunter werden die Datenelemente in der im vorherigen Kapitel entworfenen Darstellung angezeigt. Durch den ausreichend vorhandenen Platz muss dieser Entwurf nicht angepasst werden.

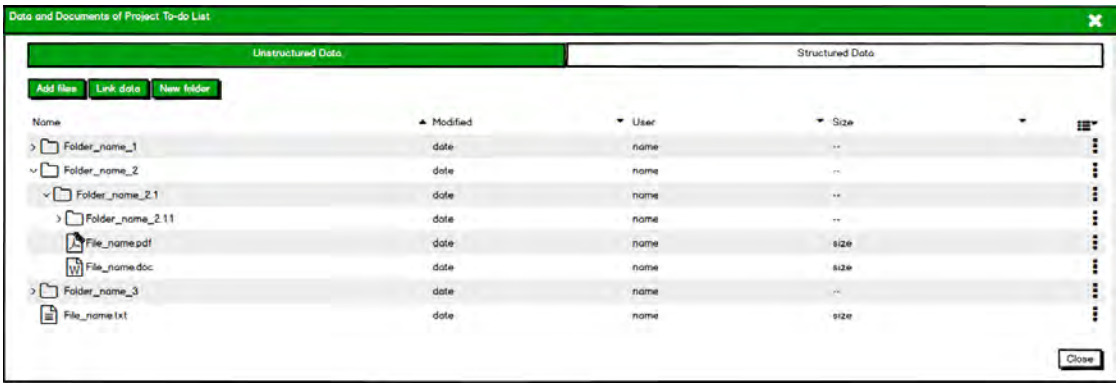


Abbildung 4.9: Modaler Dialog zur Darstellung der Datenelemente

„Data and Documents“-Ansicht

Bei dem Entwurf dieser Ansicht spielt sich der Vorteil des zuvor entwickelten modalen Dialogs zur Darstellung der Datenelemente als Baumstruktur voll aus. Denn wie in Abbildung 4.10 zu sehen ist, kann dieser Entwurf nahezu direkt für die „Data and Documents“-Ansichten übernommen werden. Einzig die Buttons müssen über den Tabs,

in der grauen Menüzeile, angebracht werden um den Darstellungsregeln der bereits vorhandenen Benutzerschnittstelle von proCollab gerecht zu werden. Es besteht nun eine nahezu identische Darstellung der Datenelemente über alle Data Spaces und unterschiedlichen Ansichten hinweg, das Ziel eine konsistente und erwartungskonforme Benutzerschnittstelle zu entwerfen wird hier erfüllt.

Abbildung 4.10: Entwurf der Ansicht „Data and Documents“

4.2.3 Input-, Output- und Hybrid-Datenelemente

Im Unterschied zu den Data Spaces, können in Task Trees die Datenelemente als Input, Output, und Hybrid deklariert werden. Die Herausforderung besteht nun darin, diese übersichtlich darzustellen und die in Abschnitt 4.2.1 entworfene Baumstruktur zur Präsentation der Datenelemente zu verwenden.

Gegenüberstellung

Die Idee bei dem in Abbildung 4.11 gezeigten Entwurf ist es, die Input- und Output-Datenelemente vertikal nebeneinander anzuordnen. Dies soll vermitteln, dass links (Input) die Voraussetzungen „hinein-“ und rechts (Output) die Ergebnisse „herauskommen“. Das hat allerdings einen Nachteil, da sich Input- und Output-Datenelemente den Platz in der Horizontalen teilen entsteht hier, ähnlich dem ersten Entwurf in Datenmanagement von Data Spaces, ein erheblicher Platzmangel. Auch haben Hybrid Datenelemente in der Vertikalen keinen Platz und müssen Horizontal darunter angebracht werden.

Abbildung 4.12 zeigt einen weiteren Entwurf, hier werden die Input-, Output- und Hybrid-Datenelemente horizontal voneinander getrennt und untereinander angeordnet. Dadurch

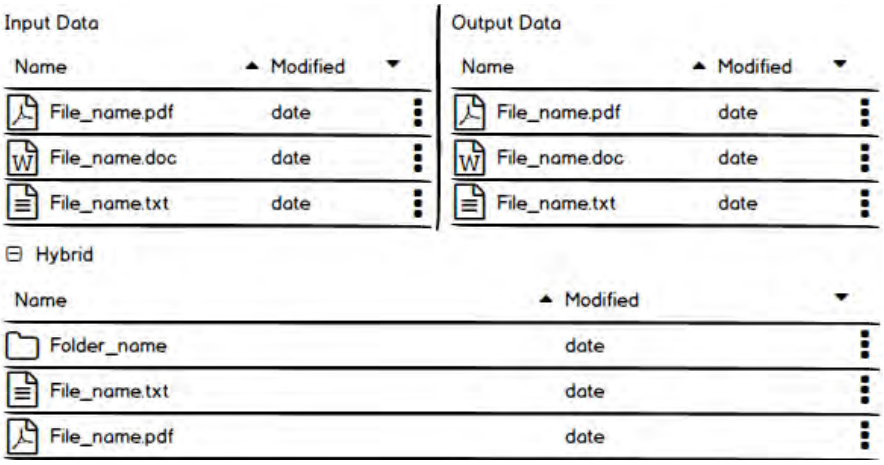


Abbildung 4.11: Vertikale Trennung der Input- und Output-Datenelemente

erübrigt sich der entstandene Platzmangel des vorherigen Entwurfs. Gleichzeitig ist diese Darstellung, trotz der Unterteilung der Datenelemente in Input, Output und Hybrid, noch sehr nahe an dem Entwurf zum Datenmanagement der Data Spaces und bedient so die Erwartungskonformität der Benutzer. Aus diesen Gründen wird letztere Idee umgesetzt.

Entwurf

Wie schon in dem in Kapitel 4.2.2 vorgestellten modalen Dialog, besitzt auch dieser Entwurf an oberster Stelle Registerkarten zum Wechseln zwischen unstrukturierten und strukturierten Daten. Der restliche Aufbau wurde leicht geändert. Um die Übersicht zu erhöhen, befinden sich die Baumstrukturen der Input-, Output- und Hybrid-Datenelemente in einem Dropdown-Menü, das mit einem Klick auf den Minus- beziehungsweise Plus-Button, in der grauen Kopfzeile ein- und ausgeklappt werden kann. Über den Button „Add files“ in der Kopfleiste lassen sich neue Datenelemente zu den jeweiligen Baumstrukturen hinzufügen. Über „Specify Input / Output / Hybrid“ können Platzhalter für zukünftige Datenelemente festgelegt werden. Dabei öffnet sich ein weiterer modaler Dialog, in dem der Benutzer den Dateinamen und Dateityp festlegen kann und diese Spezifikation daraufhin den Datenelementen hinzufügen. Anschließend kann über den Menü-Button ein Dialog zum Hochladen eines, zu einer Spezifikation passenden, Datenelements geöffnet werden.

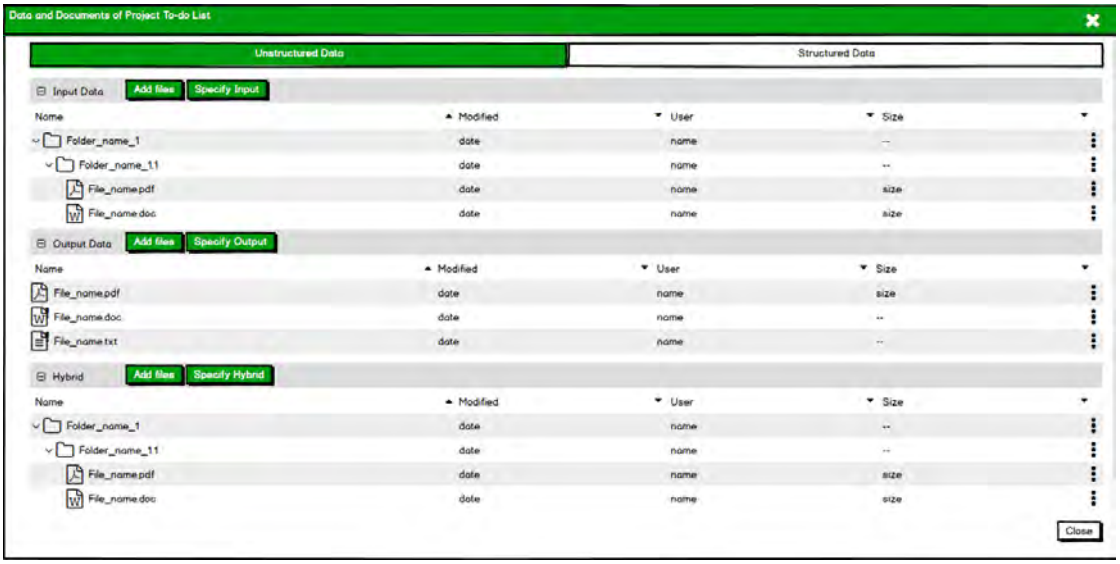


Abbildung 4.12: Horizontale Trennung der Input- und Output-Datenelemente

4.2.4 Datenübersicht in Prozessinstanz und Prozess Template

Um in der Übersicht zu geöffneten Prozess Templates und Prozessinstanzen einen schnellen Überblick über die zuletzt hinzugefügten Datenelemente zu bekommen, befindet sich zentral bei den Dropdown-Boxen eine Box „Data and Documents“. In dieser Box werden die Datenelemente angezeigt, die zuletzt dem Prozess hinzugefügt wurden (siehe Abbildung 4.12). Da in dieser Ansicht ausschließlich die neuesten Datenelemente, aber keine Ordner, angezeigt werden, besitzt die Baumdarstellung keine Knoten und es entsteht somit eine einfache Liste von Datenelementen, die dadurch weniger Platz in der Horizontalen und Vertikalen benötigt. Das Weglassen von zusätzlichen Spalten reduziert den Platzbedarf weiter und genügt trotz des Wegfalls von Informationen noch dem Anspruch der Benutzer sich einen schnellen Überblick über die zuletzt hinzugefügten Datenelemente zu verschaffen. Über den Menü-Button kann mit den Datenelementen interagiert werden - es stehen die Operationen heruntergeladen, umbenennen und löschen zur Verfügung.

4.2.5 Verwaltung strukturierter Daten Templates

Um strukturierte Daten verwenden zu können, müssen erst einmal Daten Templates, die eine strukturierte Datei definieren, erstellt werden. Dazu bietet die Benutzeroberflä-

Abbildung 4.13: Datenübersicht in der Projektübersicht

che von proCollab aktuell noch keine Möglichkeit. Nachfolgend werden Konzepte zur Darstellung der strukturierten Daten Templates und deren Verwaltung vorgestellt.

Darstellung strukturierter Daten Templates

proCollab erlaubt das Erstellen von Prozess Templates und Instanzen. Diese werden in einer Übersicht in Container dargestellt. Diese Container werden so auch für die strukturierten Daten Templates verwendet. Abbildung 4.14 zeigt das Fenster „Structured Data Templates“ zur Verwaltung der strukturierten Daten. Darin befinden sich zwei mögliche Darstellungen dieser Templates. Der linke Container ist nach dem vorhandenen Schema von Containern in proCollab erstellt worden. Im Kopf, mit grauem Hintergrund, befindet sich der Bezeichner des Containers, hier „Structured Data Template“. Darunter beginnt der Container-Inhalt mit der Bezeichnung des strukturierten Daten Templates in großer Schrift. Es folgen anschließend Metainformationen zu diesem Template. In beiden Entwürfen befinden sich rechts unten Buttons zum Bearbeiten und Löschen des strukturierten Daten Templates.

Der zweite Entwurf hält sich nicht vollständig an das Schema der sonstigen Container. Im Kopf des Containers befindet sich nun nicht mehr der Container Bezeichner, da der Benutzer davon ausgehen kann, dass sich im Menüpunkt „Structured Data Templates“ eben genau diese Templates befinden. Stattdessen ist im Kopf der Bezeichner des Templates untergebracht, so wird im Container-Inhalt einiges an Platz für zusätzliche Informationen frei. Im Inneren des Containers werden die ersten Attribute und die dazu-

gehörenden Datentypen angezeigt um so eine Vorschau auf den Inhalt des strukturierten Daten Templates zu zeigen.

Trotz einiger Vorteile des rechten Entwurfs, wurde der linke ausgewählt. Das Ziel einer konsistenten und erwartungskonformen Benutzererfahrung ist höher zu gewichten als die gewonnene Übersicht des rechten Entwurfs.

Abbildung 4.14: Zwei Gestaltungsvarianten von strukturierten Daten Templates.

Erstellen und Bearbeiten von strukturierten Daten Templates

Das in Abbildung 4.15 gezeigte Pop-up erscheint nach einem Klick auf das Hinzufügen-Symbol in der Ansicht "Strukturierte Daten verwalten". Dieser Dialog erlaubt dem Benutzer für das neue strukturierte Daten Template einen Namen zu vergeben und diesem beliebig viele Attribute, mit unterschiedlichen Datentypen aus einem Dropdown-Menü, hinzuzufügen. Weitere Attributsfelder lassen sich über den „+ Add Attribute“ Link hinzufügen. Der Button "Add" erstellt das Template und es erscheint in der Übersicht "Strukturierte Daten verwalten".

Soll ein strukturiertes Daten Template geändert werden (siehe Abbildung 4.15 unten), erscheint ebenfalls ein Pop-up mit selben Aufbau, wie das zum Erstellen neuer strukturierter Daten Templates. Hier lassen sich die schon ausgefüllten Attributsfelder und der Templatebezeichner ändern oder weitere Attributsfelder hinzufügen. Über den Button "Update" lassen sich vorgenommene Änderungen übernehmen.

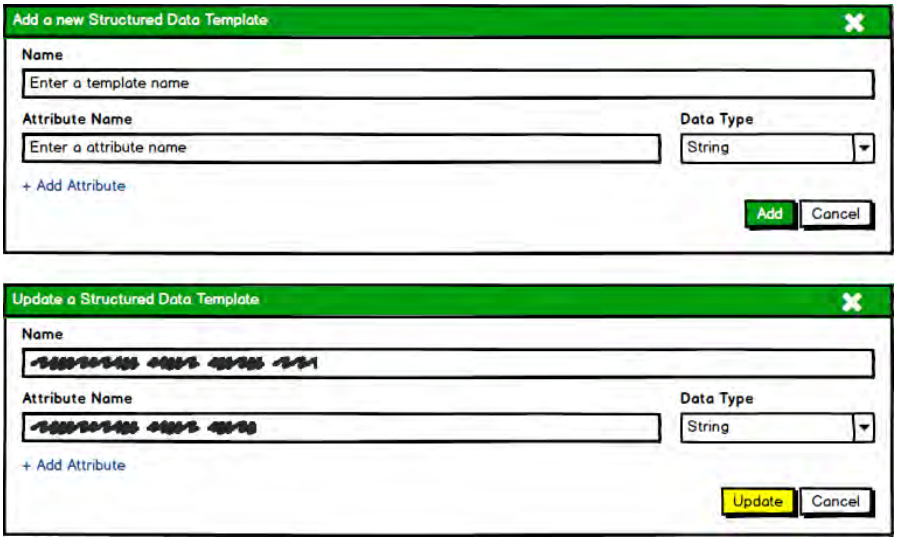


Abbildung 4.15: Hinzufügen und Ändern strukturierter Daten Templates

5

Realisierung

In Kapitel 5 werden ausgewählte UI-Mockups in den proCollab Web-Client implementiert. Doch bevor dies geschehen kann, wird in Kapitel 5.1 ein Überblick über die im Web-Client verwendeten Technologien gegeben. Anschließend folgt in Kapitel 5.2 die Betrachtung der Projektstruktur und den benötigten Services. In Kapitel 5.3 folgt schließlich die Implementierung der UI-Mockups in den proCollab Web-Client.

5.1 Verwendete Technologien

Zur Implementierung der UI-Mockups in den proCollab Web-Client werden die Technologien *Angular 4*, *TypeScript* und *Sass* benötigt. Diese werden in den folgenden Unterkapiteln vorgestellt.

5.1.1 Angular 4

Angular 4 ist eine Entwicklungsplattform und Framework, um plattformübergreifende Webanwendungen in HTML und TypeScript zu entwickeln [Goo18]. Dabei setzt Angular 4 auf einen stark modulierten Aufbau (Siehe Abbildung 5.1). Diese Module enthalten Code für zusammengehörende Funktionen einer Webanwendung, wie Komponenten oder Services. Jede Anwendung in Angular besitzt mindestens eine Komponente, die die Programmlogik enthält. Zusammen mit einem HTML-Template definieren sie eine Ansicht, die auf dem Zielsystem dargestellt wird. Jede Komponente wird durch Metadaten mit einem HTML-Template verknüpft. Metadaten sind Informationen für Angular, die die Beziehungen zwischen den beiden Teilen abbilden.

Das HTML-Template enthält neben HTML auch Angular Code und wird dadurch manipulierbar. Dazu stehen in Angular Direktiven (z.B. **ngIf* und **ngFor*) zur Verfügung, die in

HTML-Elemente integriert werden. Sind eine Komponente und ein HTML Template miteinander verknüpft, kann die Webanwendung durch Event-Binding auf Nutzereingaben reagieren und durch Property-Binding Änderungen an der HTML-Ansicht vornehmen.

Services werden genutzt, um wiederverwendbaren Code bereitzustellen. Mit *Dependency Injection* können diese beliebig vielen Komponenten zur Verfügung gestellt werden.

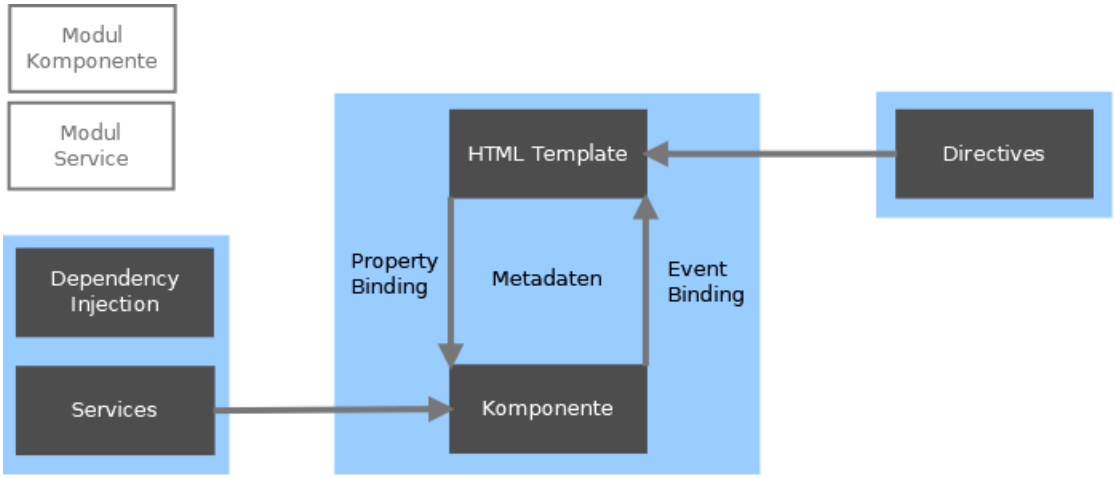


Abbildung 5.1: Zusammenhang der Hauptbestandteile von Angular

5.1.2 TypeScript

TypeScript ist eine, von Microsoft entwickelte, Open Source Programmiersprache, die auf JavaScript aufbaut und diese erweitert. Eine dieser Erweiterungen ist die Typisierung von Datentypen. Während in JavaScript einer Variablen dynamisch ein Datentyp zugewiesen wird der sich dem Wert der Variablen anpasst, kann Variablen in TypeScript ein fester Datentyp (z.B. String, Number oder Boolean) zugewiesen werden [Mic18]. Dies hat zur Folge, dass in TypeScript die Variablen nur Werte annehmen können, die diesem Datentyp entsprechen.

Da JavaScript eine Teilmenge von TypeScript ist, funktionieren JavaScript Programme auch mit TypeScript. Ein Compiler übersetzt den TypeScript Code in JavaScript Code, der dann wiederum von Webbrowsern interpretiert werden kann.

5.1.3 Sass

Wie bei TypeSkript zu JavaSkript ist *Sass* (Syntactically Awesome Stylesheets) eine Übersparache zu CSS (Cascading Style Sheets). Da CSS eine Untermenge von Sass ist, ist jeder CSS-Code auch Sass-Code. Durch die erweiterte Syntax, genannt *SCSS*, hat Sass die Vorteile, dass sich der Code übersichtlicher und besser strukturieren lässt, als reiner CSS-Code. Darüber hinaus werden auch die Gestaltungsmöglichkeiten erweitert [Sas18].

5.2 Ausgangssituation

Der proCollab Web-Client hat, in seinem aktuellen Stand, eine gewisse Komplexität erreicht. Deshalb ist eine ausführliche Auseinandersetzung mit der Projektstruktur notwendig um sich im Code des Web-Clients zurechtzufinden. Es soll aufzeigen wo sich die für die Implementierung der UI-Mockups benötigten *Verzeichnisse*, *Komponenten* und *Services* (siehe 5.1.1) befinden.

Projektstruktur

Abbildung 5.2 zeigt einen Auszug der für dieses Projekt wichtigsten Verzeichnisse aus der Projektstruktur des proCollab Web-Client. Alle relevanten Verzeichnisse befinden sich im *app-Verzeichnis*. Im *dashboard-Verzeichnis* befinden sich die Komponenten für die Hauptansichten des Web-Clients. Jede Komponente besitzt eine index.ts Datei, die alle Dateien der Komponente nach außen hin verfügbar macht. Sowie eine HTML-Datei die, gerendert wird und einer Komponenten-Datei die die Logik enthält.

Weiter befindet sich im *app-Verzeichnis* der Ordner *service* dort werden Funktionen die in die Komponenten eingebunden werden bereitgestellt. Ein wichtiger Service wird im nächsten Unterabschnitt vorgestellt. Die Verzeichnisse *shared* und *widgets* enthalten ebenfalls Komponenten. Das *shared-Verzeichnis* stellt Komponenten, wie die Menüzeile oder die Brotkrümelnavigation, die in vielen Ansichten benötigt werden zur Verfügung. Der Ordner *widgets* hält Elemente die in jede Ansicht eingebunden werden können bereit. Dies können modale Dialoge sein oder Dropdown-Menüs.

Im *assets-Verzeichnis* sind Bilddateien und Icons untergebracht. Die für die Gestaltung der Ansichten notwendigen SCSS-Dateien (siehe Kapitel 5.1) befinden sich im *css-Verzeichnis*. Auf alle Regelsätze, die hier in SCSS-Dateien organisiert und registriert sind, stehen im gesamten Projekt zur Verfügung und können in die Komponenten eingebunden werden.

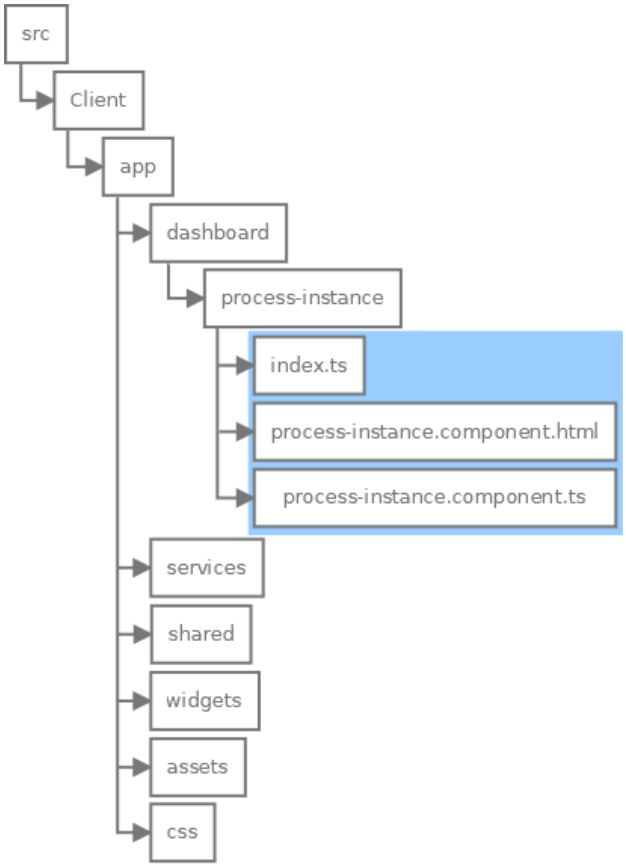


Abbildung 5.2: Projektstruktur

EventEmitter

Durch die starke Modulierung von Angular-Anwendungen, ist es häufig der Fall, dass eine Ansicht aus mehreren verschiedenen Komponenten gebildet wird. Dadurch ist es wichtig, dass die Komponenten über Änderungen in anderen Komponenten informiert werden können. In Angular existiert zu diesem Zweck ein EventEmitter.

Dieser Service kümmert sich um die Kommunikation zwischen Kind- und Elternkomponente. In der Kindkomponente muss in einer Methode festgelegt werden welche Informationen zur Elternkomponente übermittelt werden sollen. In der Elternkomponente, die den EventEmitter abonniert, können diese Informationen in einer Methode entgegengenommen und direkt verarbeitet werden.

Modaler Bestätigungsdialog

Vor dem Ausführen bestimmter Aktionen durch den Benutzer, kann dieser aufgefordert werden die Durchführung nochmals zu bestätigen. Dafür steht im Web-Client die Komponente *ConfirmationModalComponent* zur Verfügung. Diese implementiert einen modalen Dialog, der den Benutzer zum Beispiel vor einem Löschvorgang warnt und bestätigt werden muss (siehe Abbildung 5.3).

Wird diese Komponente in einer Elternkomponente implementiert, kann das Aussehen und der Inhalt des Bestätigungsdialogs festgelegt werden. Dazu werden Attribute wie Farbe, Dialogtitel und Art des Buttons in einem Objekt gespeichert und über den EventEmitter an die Kindkomponente übermittelt. Wird der Bestätigungsdialog geschlossen, benachrichtigt der Service die Elternkomponente über die Änderungen.

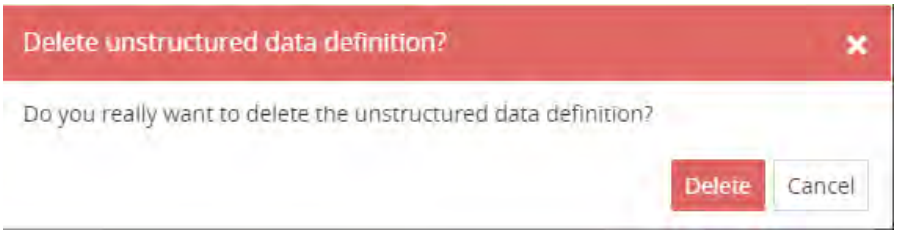


Abbildung 5.3: Beispiel eines modalen Bestätigungsdialogs

5.3 Implementierung

Um die Umsetzbarkeit der UI-Entwürfe zu bestätigen, werden in diesem Abschnitt die „Data and Documents“-Ansicht, sowie der modale Dialog für Input-, Output und Hybrid-Datenelemente, in den proCollab Web-Client implementiert. Die Implementierungen sollen es ermöglichen Metainformationen (z.B. Dateiname, Änderungsdatum) zu Daten-

elementen anzuzeigen, strukturierte Daten anzulegen und zu bearbeiten ebenso sollen Spezifikationen für Input-, Output und Hybrid-Datenelemente festgelegt werden können.

5.3.1 Datenmanagement von Data Spaces

In Kapitel 3.2.1 wurden Aufgaben definiert die der Benutzer am Datenmanagement von Data Spaces durchführen muss. Aus diesen Aufgaben werden, das Hinzufügen und Anzeigen von Datenelementen, das Hinzufügen und Bearbeiten von strukturierten Daten und das Entfernen von Datenelementen in die „Data and Documents“-Ansichten implementiert.

Hinzufügen unstrukturierter Daten

Im proCollab Web-Client existiert eine Komponente, die eine graue Menüzeile, mit Buttons und Bezeichner, in den oberen Rand der Inhaltsansicht integriert. Diese Buttons werden nicht aus dem häufig gebrauchten HTML-Element *<button>* erzeugt, sondern sind Flächen die auf ein Klick-Event hören und den Identifikationswert der gedrückten Fläche an eine Methode weiterreichen, die sich um dieses Klick-Event kümmert. Um unstrukturierte Daten hochladen zu können, wird, allerdings ein Input-Button benötigt, der den Dateiselektor des Betriebssystems öffnet. In der Menüzeile werden deswegen, statt der in der Komponente vorgesehenen Buttons, HTML-Elemente verwendet.

Da sich Input-Elemente nicht gut mit CSS-Regelsätzen anpassen lassen, wird zusätzlich noch das HTML-Element *<label>* genutzt. Um einen Button zu erhalten, der die Funktion eines HTML Input-Elements und das Aussehen der üblichen in proCollab genutzten Buttons hat, wird wie folgt vorgegangen.

Der Input-Button wird durch einen CSS-Regelsatz auf 0,1 Pixel geschrumpft, so ist er für den Benutzer unsichtbar, wird aber vom Browser dennoch gerendert. Das Label nimmt mit den vorhandenen CSS-Regelsätzen die Gestalt der üblichen Buttons an und wird über eine *ID* mit dem Input-Element verbunden. Zusätzlich wird über eine *Expression* der Dateiname, der ausgewählten Datei auf dem Label angezeigt. Sobald eine Datei ausgewählt wurde, erscheint in der Menüzeile ein Upload-Button, über den die ausgewählte Datei hochgeladen werden kann.

Metainformationen zu Unstrukturierte Daten

Im proCollab Web-Client ist bereits ein Dateibaum in die „Data and Documents“-Ansichten implementiert. Allerdings werden keine Metainformationen zu den unstrukturierten Daten und Ordner angezeigt. Auch kann dieser nicht, durch zusätzliche Spalten, erweitert werden. Um dennoch Metainformationen anzuzeigen wird neben dem Dateibaum eine Tabelle aus *div-Elementen* erzeugt. Dazu stehen die CSS-Eigenschaften *display: table, table-row* und *table-cell* zur Verfügung. Die div-Elemente verhalten sich anschließend so wie ein HTML Tabellen-Element.

In der Komponente der „Data and Documents“-Ansicht werden die Elemente des Dateibaums in das Array *unstructuredFiles[]* geschrieben. In Zeile 2 des Listing 5.1 wird im HTML-Template die Tabelle aus div-Elementen so implementiert, dass die Direktive **ngFor = "let file of unstructuredFiles"* für jedes Array-Element eine neue Tabellenzeile erzeugt. Anschließend kann in den Tabellenspalten (siehe Zeile 3 - 11) mit Hilfe von Angular-Expressionen, die gewünschte Metainformation aus der Serverantwort extrahiert werden (siehe Abbildung 5.4).

```
1  <div class="table-div unstr-table">
2    <div *ngFor = "let file of unstructuredFiles" class="table-row
       -div">
3      <div class="div-cell">{{file.dateUpdated | date: "dd.MM.y HH:
         mm"}}</div>
4      <div class="div-cell">{{file.creatorName}}</div>
5      <div class="div-cell">{{file.fileSize}} bytes</div>
6      <div class="div-cell">
7        <div class="pull-right">
8          <data-management-dropdown [tab]="tab">
9          </data-management-dropdown>
10       </div>
11     </div>
12   </div>
13 </div>
```

Listing 5.1: Auszug der Tabelle für Metainformationen

Über einen Button, der in der letzten Spalte jeder Zeile vorhanden ist, kann ein Dropdown-Menü aufgerufen werden. Hierüber sind die wichtigsten Funktionen (z.B. Download, Um-

benennen, Löschen) aufrufbar. Das Dropdown-Menü ist eine eigenständige Komponente und wird als HTML-Element *<data-management-dropdown ...></data-management-dropdown>* in die „Data and Documents“-Ansicht integriert.

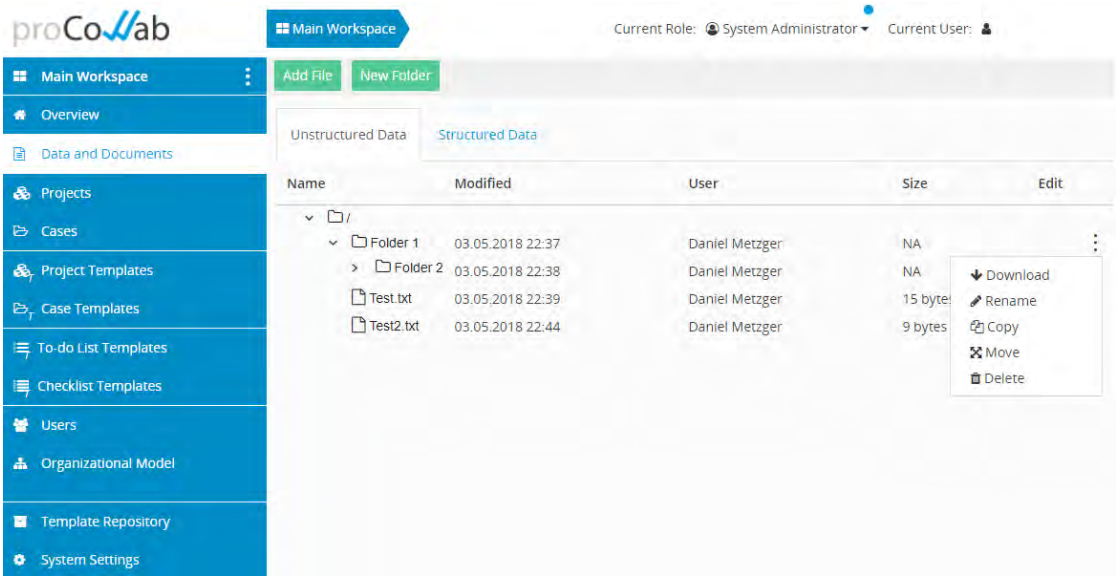


Abbildung 5.4: Unstrukturierte Daten in der „Data and Documents“-Ansicht

Registerkarten

Um zwischen unstrukturierten und strukturierten Daten zu wechseln, werden Registerkarten benötigt. Für die Registerkarten wird eine vorhandene Implementierung des Web-Clients genutzt, dadurch wird ein konsistentes User-Interface sichergestellt. In der Komponente der „Data and Documents“-Ansicht wird die Variable *tab* mit dem Wert *unstructured* initialisiert. In dem HTML-Template wird, mit der Angular Direktive **ngIf* der Wert der Variablen *tab* verglichen, ist er *unstructured* werden die unstrukturierten Daten angezeigt, ist er *structured* werden die strukturierten Daten angezeigt.

Strukturierte Daten

Strukturierte Daten können nicht in Order abgelegt werden, deshalb ist hier kein Dateibaum notwendig. Eine einfache Tabelle genügt, um die strukturierten Daten mit den dazugehörigen Metainformationen anzuzeigen (siehe Abbildung 5.5). Über das

Dropdown-Menü, welches erscheint wenn auf den Button mit den drei vertikal angeordneten Punkten, am Ende einer Zeile, geklickt wird, kann ein strukturiertes Datenelement geändert werden.

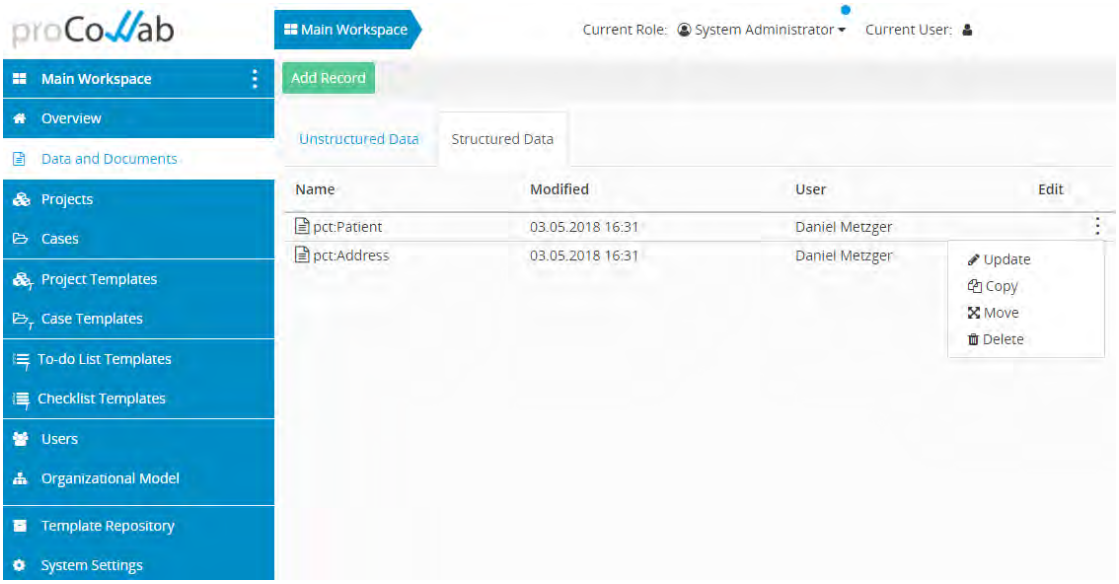


Abbildung 5.5: Strukturierte Daten in der „Data and Documents“-Ansicht

Zwischen dem Aufrufen des Dropdown-Menüs und dem Schließen des modalen Dialogs zum Ändern strukturierten Daten, findet ein Informationsfluss zwischen den beteiligten Komponenten statt. Die Komponente der „Data and Documents“-Ansicht ist die Elternkomponente, während das Dropdown-Menü und der modale Dialog zum Ändern der strukturierter Daten die Kinderkomponenten sind. Wird nun das Dropdown-Menü geöffnet, wird ein Objekt, das die strukturierte Datei beschreibt, von der Elternkomponente übergeben. Klickt der Benutzer nun auf *Update*, übermittelt die Dropdown-Komponente dieses Objekt, über den EventEmitter, zurück an die Elternkomponente. Die Komponente der „Data and Documents“-Ansicht hört schon seit ihrer Initialisierung auf dieses Ereignis und nimmt das Objekt durch den EventEmitter entgegen und gibt es direkt an die Methode *editStructuredDataTemplate()* weiter (siehe Listing 5.2). In dieser Methode wird nun der modale Dialog zum Ändern der strukturierten Datei aufgerufen. Dieser Dialog ist HTML-Template der „Data and Documents“-Ansicht als HTML-Element *<edit-structured-data-modal ...></edit-structured-data-modal>* eingebunden und übermittelt die ID des genutzten Workspace und das Objekt an den modalen Dialog.

```
1 subscribe() {
2 EmitterService.get('editStructuredDataTemplate').subscribe((
    data:any) => {
3   this.editStructuredDataTemplate(data.structuredDataTemplate);
4    });
5 }
6 editStructuredDataTemplate(structuredDataTemplate:
    StructuredDataTemplateDto) {
7   this.editTemplate = structuredDataTemplate;
8   this.showEditStructuredDataModal = true;
9 }
```

Listing 5.2: Aufruf eines modalen Dialogs durch den Emitter Service

Wie in Kapitel 4.1.1 beschrieben existieren für modale Dialoge bereits Darstellungsregeln. Diese werden hier angewendet, so stimmen Form und Position mit den vorhandenen modalen Dialogen des Web-Client überein. Abbildung 5.6 zeigt den modalen Dialog zum Ändern von strukturierten Daten. Die Eingabefelder sind mit den Daten gefüllt die dem Dialog durch das übermittelte Objekt bereitgestellt werden. Der Benutzer hat nun die Möglichkeit diese zu ändern. Der Button *Update* ruft eine Methode auf die diese Änderungen an den Server übermittelt und anschließend über den EventEmitter der Elternkomponente mitteilt, dass der modale Dialog geschlossen werden soll.

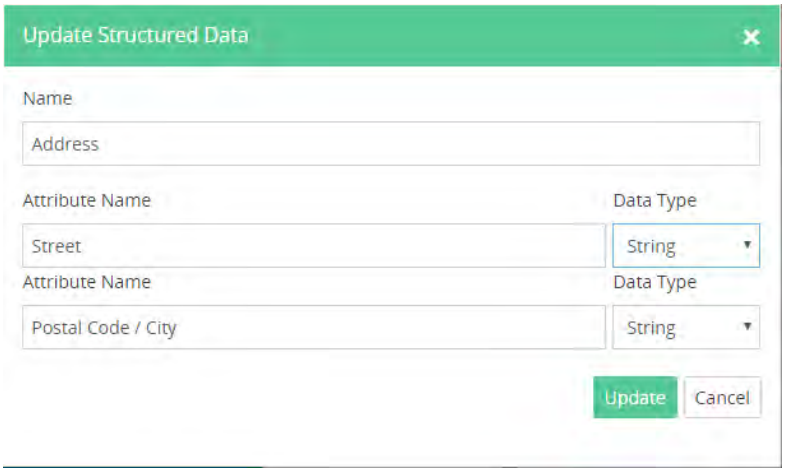


Abbildung 5.6: Implementierter modaler Dialog zur Änderung strukturierter Daten

5.3.2 Modaler Dialog für Input-, Output- und Hybrid-Datenelemente

Die Implementierung der Registerkarten, um zwischen unstrukturierten und strukturierten Daten zu wechseln, stimmt mit der in Kapitel 5.3.1 überein. Neu ist hier jedoch, dass für Input, Output und Hybrid jeweils ein separater Dateibaum dargestellt werden soll. Dieser ist in einem Dropdown-Container untergebracht. Jeder Container enthält eine graue Menüzeile, die alle Buttons, das Plus- bzw. Minussymbol zum ein- und ausklappen des Containers und einen Bezeichner am rechten Rand beinhaltet (siehe Abbildung 5.7). Wird ein Dropdown-Container eingeklappt, ändert sich das Minussymbol zu einem Plussymbol und der Containerinhalt sowie die Buttons in der Menüzeile werden ausgeblendet.

Um nicht für Input, Output und Hybrid jeweils einen eigenen Dropdown-Container in HTML zu implementieren, wird dieser nur einmal in das HTML-Template implementiert und mit den Angular-Direktiven **ngFor* und **ngIf* versehen. In der Komponenten-Datei wird ein Array erstellt, das mit drei Objekten gefüllt wird. Jedes Objekt beschreibt eines der Datenelementzuordnungen Input, Output oder Hybrid. Zur Laufzeit wird dann aus dem HTML-Code durch die Angular-Direktiven für jedes Array-Element ein eigener Dropdown-Container erstellt. Jeder Dropdown-Container erhält dabei eine eigene Bezeichnung für Input-, Output- und Hybrid-Datenelemente und lässt sich auch unabhängig von den anderen Containern ein-, beziehungsweise ausklappen. Für die Anzeige der Datenelemente wird die in Kapitel 5.3.1 implementierte Tabelle verwendet.

Wie jeder andere modale Dialog, wird auch dieser geschlossen, indem die Elternkomponente durch den EventEmitter darüber informiert wird und diese den Dialog daraufhin schließt.

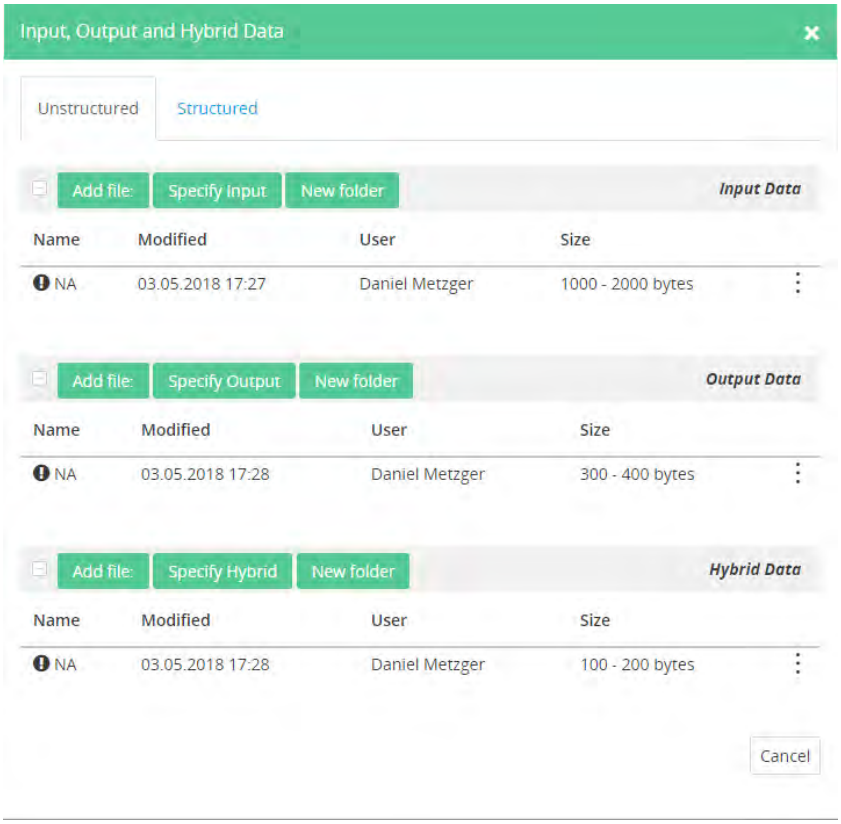


Abbildung 5.7: Modaler Dialog für Input-, Output- und Hybrid-Datenelemente

Spezifikation für Input-, Output oder Hybrid-Datenelemente hinzufügen

Um die Tabellen in den Dropdown-Containern füllen zu können, wird ein weiterer modaler Dialog als neue Komponente erstellt, die dem modalen Dialog zum Ändern strukturierter Daten (siehe Kapitel 5.3.1) ähnelt. Darin kann der Benutzer eine Spezifikation für Input-, Output- oder Hybrid-Datenelemente erstellen (siehe Abbildung 5.8).

Alle Spezifikationen werden in einem Array gespeichert, eine Unterscheidung zwischen Input-, Output- und Hybrid-Datenelemente findet nicht statt. Damit in den jeweiligen Dropdown-Containern die richtigen Spezifikationen angezeigt werden, wird die in Listing 5.1 vorgestellte Implementierung erweitert. Der *ngFor Schleife in Zeile 2 wird eine *Pipe* hinzugefügt. Pipes ermöglichen es in Angular 4 Daten vor dem Rendern zu transformieren oder zu formatieren. Diese Pipe zeigt nur die Spezifikationen aus dem Array *unstructuredDefinitions* an, die der Zuordnung Input, Output oder Hybrid des jeweiligen Dropdown-Containers entsprechen.

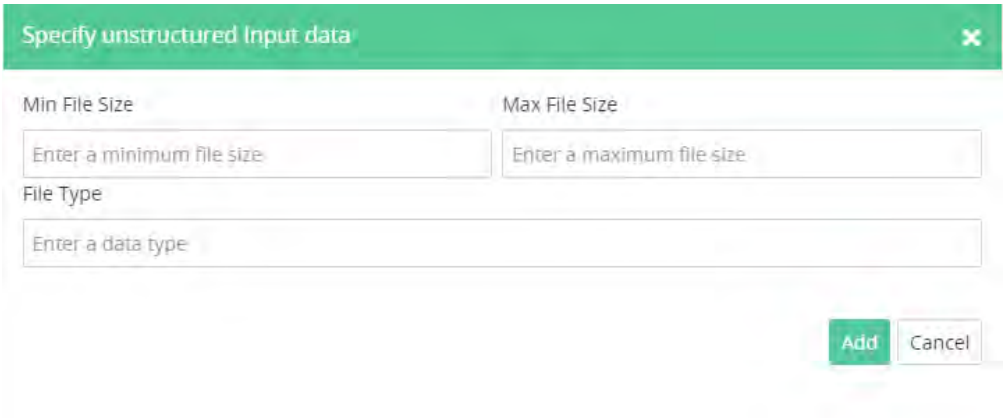


Abbildung 5.8: Spezifikation von Input-, Output- und Hybrid-Datenelemente

Da in Angular kein Pipe existiert, die diese Funktion erfüllt, wird eine eigene Pipe erstellt (siehe Listing 5.3). Wird die Pipe mit dem Namen *inputOutputFilter* aufgerufen, nimmt die *transform()* Methode in Zeile 3 ein Array mit Spezifikationen von unstrukturierten Input-, Output- und Hybriddatenelementen entgegen. Der Variablen *term* wird, entsprechend des Dropdown-Containers der die Pipe aufruft, der Wert *Input*, *Output* oder *Hybrid* zugewiesen. In Zeile 5 wird für jedes Element des Arrays *unstructuredDefinition* die JavaScript Methode *filter()* aufgerufen und in der Variable *definition* übergeben. In Zeile 6 wird verglichen, ob das Attribut *dataflowType* dem übergebenem *term* entspricht. Trifft dies zu wird das Element im *unstructuredDefinition* Array behalten, ansonsten wird es entfernt.

```
1  @Pipe({name: 'inputOutputFilter'})
2  export class InputOutputFilterPipe implements PipeTransform {
3    transform(unstructuredDefinition: any[], term: any) {
4      if(unstructuredDefinition) {
5        return unstructuredDefinition.filter(function(definition)
             {
6          return definition.dataflowType === term;
7        });
8      }
9      return null;
10   }
11 }
```

Listing 5.3: Pipe um Input-, Output- und Hybrid-Datenelemente zu Filtern

6

Fazit

Im letzen Kapitel dieser Arbeit wird zusammengefasst was in den vorangegangenen Kapiteln geleistet wurde, dabei wird auch auf Probleme eingegangen, die während der Arbeit auftraten und welche Schlüsse daraus gezogen wurden. Abschließend folgt ein kurzer Ausblick auf mögliche nachfolgende Projekte.

6.1 Zusammenfassung

Das Ziel dieser Arbeit war es ein UI-Konzept für das proCollab Datenmanagement zu entwickeln. Kapitel 1 führte dazu in die Problematik ein und erläutert meinen Beitrag für diese Arbeit.

In Kapitel 2 wurde in das Thema Wissensintensive Prozesse eingeführt und das grundlegende Konzept von proCollab und dem Data Repository beschrieben. Abschließend wurde Usability definiert und die Grundsätze der Dialoggestaltung vorgestellt, die später in das UI-Konzept mit einflossen.

Als Basis für den späteren UI-Mockup Entwurf wurden in Kapitel 3 die grundlegenden Anforderungen an das UI-Konzept definiert. Dafür wurde dar aktuelle Stand des proCollab Projektes, mit besonderem Fokus auf das Datenmanagement festgehalten. Zusätzliche Erkenntnisse und Ideen für das zu entwickelnde UI-Konzept lieferte der Vergleich von ähnlichen Systemen. Im Rahmen der Aufgabenanalyse wurde festgehalten welche Funktionen der Benutzer am System durchführen können muss.

Auf diesen Informationen aufbauend entstand in Kapitel 4 schließlich das UI-Konzept. Dabei wurden zunächst, im Rahmen des konzeptuellen UI-Modells, grundlegende Darstellungsregeln für die Gestaltung der UI-Mockups erarbeitet. Anschließend wurden verschiedene Mockup-Entwürfe gemäß den Anforderungen entworfen und diskutiert. Dabei war die Herausforderung das UI-Konzept so zu entwickeln, dass es sich konsistent

in das schon vorhandne User-Interface integriert. Der Benutzer soll nicht den Eindruck gewinnen, dass es erst nachträglich hinzugefügt wurde, sondern seine aus den bisherigen Erfahrungen mit dem bestehenden User-Interface entstandenen Erwartungen erfüllt.

Bevor die Implementierung in Kapitel 5 durchgeführt werden konnte, musste zunächst eine Einarbeitung in die verwendeten Technologien des Web-Clients stattfinden. Als vorletzter Schritt musste ein Überblick über die bereits vorhandene Projektstruktur, sowie den wichtigsten Funktionen des proCollab Web-Clients gewonnen werden. Zum Abschluss dieser Arbeit wurde die Umsetzbarkeit mit der Implementierung ausgewählter UI-Mockups gezeigt.

6.2 Ausblick

Implementierung

Das Kapitel 5 hat gezeigt, dass das entworfene UI-Konzept umsetzbar ist. Als weiterführende Arbeit können die implementierten UI-Mockups in allen Ansichten untergebracht werden für die sie vorgesehen sind. Weiter können die vorhandenen Ansichten um die noch nicht umgesetzten UI-Mockups erweitert werden und somit den in Kapitel 3.2 definierten Aufgaben gerecht zu werden.

Erweiterungen

Das UI-Konzept kann dahingehend erweitert werden, dass es eine Versionierung von Datenelemente unterstützt. Wissensarbeiter sollen die Möglichkeit haben Datenelemente durch eine neuere Version derselben zu ersetzen. Das User Interface muss dabei die Version des Datenelements hervorheben und es ermöglichen vorangegangene Versionen anzuzeigen und Änderungen wieder rückgängig zu machen.

Eine weitere nützliche Funktion ist das Hinzufügen von Notizen an Datenelemente. Dies soll zum zusätzlichen Austausch unter Wissensarbeitern dienen, um unter anderem Hinweise oder Anmerkungen über das betroffene Datenelement, als zusätzliche Information bereitzustellen.

Tabellenverzeichnis

3.1 Elemente eines Flussdiagramms 28
3.2 Meistgenutzte Desktopauflösung Januar 2018 [Sta18b, W3S18a] 54
3.3 Marktanteile von Desktop Webbrowser im Januar 2018 [Sta18a, W3S18b] 55

Abbildungsverzeichnis

2.1 Der Lebenszyklus wissensintensiver Prozesse [MKR12] 9
2.2 proCollab Übersicht [MR17] 11
2.3 proCollab Datenmanagement 15

3.1 Hauptansicht des proCollab Web-Client bei geöffneter Projektinstanz . . . 21
3.2 Asana 23
3.3 Docs & Files von Basecamp 24
3.4 Wrike 25
3.5 Dropbox 26
3.6 Use-Case Diagramm zum Datenmanagement von Data Spaces 29
3.7 Flussdiagramm: Datenelemente anzeigen 29
3.8 Flussdiagramm: Unstrukturierte Daten hinzufügen 30
3.9 Flussdiagramm: Strukturierte Daten hinzufügen 31
3.10 Flussdiagramm: Datenelement verlinken 32
3.11 Flussdiagramm: Datenelemente kopieren / verschieben 33
3.12 Flussdiagramm: Datenelemente sortieren 34
3.13 Flussdiagramm: Unstrukturierte Daten herunterladen 35
3.14 Flussdiagramm: Strukturierte Daten bearbeiten 36
3.15 Flussdiagramm: Ordner erstellen 37
3.16 Flussdiagramm: Datenelemente und Ordner löschen 38
3.17 Flussdiagramm: Datenelemente und Ordner umbenennen 39
3.18 Spezifizierte Input-, Output- und Hybrid-Datenelemente 40
3.19 Flussdiagramm: Input, Output und Hybrid spezifizieren 41
3.20 Flussdiagramm: Spezifizierung bearbeiten 42
3.21 Flussdiagramm: Spezifizierung löschen 43
3.22 Flussdiagramm: Spezifizierte Datei hochladen 44
3.23 Use-Case Diagramm zur Datenübersicht in Prozessinstanz und Prozess Template 45
3.24 Flussdiagramm: Unstrukturierte Daten herunterladen 46
3.25 Flussdiagramm: Strukturierte Datei bearbeiten 47
3.26 Flussdiagramm: Datenelement löschen 48
3.27 Flussdiagramm: Datenelemente sortieren 49
3.28 Use-Case Diagramm zur Verwaltung strukturierter Daten 49

3.29 Flussdiagramm: Strukturierte Daten Templates anzeigen 50
3.30 Flussdiagramm: Strukturierte Daten Templates erstellen 51
3.31 Flussdiagramm: Strukturierte Daten Templates ändern 52
3.32 Flussdiagramm: Strukturierte Daten Templates löschen 53

4.1 Inhaltsbereich 58
4.2 Beispiel eines modalen Dialogs 59
4.3 Beispiel eines Containers 59
4.4 Dialogstruktur 60
4.5 Entwurf einer Liste zur Darstellung von Datenelementen 61
4.6 Entwurf: Baumdarstellung der Datenelemente 63
4.7 Thumbnailansicht von unstrukturierten Daten 64
4.8 Projektübersicht mit expandierter To-do List 65
4.9 Modaler Dialog zur Darstellung der Datenelemente 66
4.10 Entwurf der Ansicht „Data and Documents“ 67
4.11 Vertikale Trennung der Input- und Output-Datenelemente 68
4.12 Horizontale Trennung der Input- und Output-Datenelemente 69
4.13 Datenübersicht in der Projektübersicht 70
4.14 Zwei Gestaltungsvarianten von strukturierten Daten Templates. 71
4.15 Hinzufügen und Ändern strukturierter Daten Templates 72

5.1 Zusammenhang der Hauptbestandteile von Angular 74
5.2 Projektstruktur 76
5.3 Beispiel eines modalen Bestätigungsdialogs 77
5.4 Unstrukturierte Daten in der „Data and Documents“-Ansicht 80
5.5 Strukturierte Daten in der „Data and Documents“-Ansicht 81
5.6 Implementierter modaler Dialog zur Änderung strukturierter Daten 82
5.7 Modaler Dialog für Input-, Output- und Hybrid-Datenelemente 84
5.8 Spezifikation von Input-, Output- und Hybrid-Datenelemente 85

Literaturverzeichnis

[Asa17] ASANA, Inc.: *Asana*. https://asana.com. Version: 10.11.2017

[Bas17] BASECAMP, L.L.C.: *Basecamp*. https://basecamp.com. Version: 10.11.2017

[DIN16] DIN EN ISO-9241-11: *Ergonomie der Mensch-System-Interaktion - Teil 11: Gebrauchstauglichkeit: Begriffe und Konzepte*. 2016

[DMR14] DI CICCIO, Claudio ; MARRELLA, Andrea ; RUSSO, Alessandro: *Knowledge-intensive processes: Characteristics, requirements and analysis of contemporary approaches*. Journal on Data Semantics, 2014

[Dro17] DROPBOX, Inc.: *Dropbox*. https://www.dropbox.com. Version: 13.11.2017

[Goo18] GOOGLE INC.: *Angular - Architecture*. https://angular.io/guide/architecture. Version: 2018

[Hub05] HUBE, Gerhard: *Beitrag zur Beschreibung und Analyse von Wissensarbeit*, Universität Stuttgart, Dissertation, 2005

[May10] MAYHEW, Deborah J.: *The usability engineering lifecycle : a practitioner's handbook for user interface design*. Morgan Kaufmann, 2010

[MBR15] MUNDBROD, Nicolas ; BEUTER, Florian ; REICHERT, Manfred: Supporting Knowledge-Intensive Processes through Integrated Task Lifecycle Support. In: IEEE COMPUTER SOCIETY PRESS (Hrsg.): *2015 IEEE 19th International Enterprise Distributed Object Computing Conference*, 2015, S. 19–28

[Mic18] MICROSOFT CORPORATION: *TypeScript Documentaton*. https://www.typescriptlang.org/docs/handbook/basic-types.html. Version: 18.01.2018

[MKR12] MUNDBROD, Nicolas ; KOLB, Jens ; REICHERT, Manfred: Towards a System Support of Collaborative Knowledge Work. In: SPRINGER (Hrsg.): *1st Int'l Workshop on Adaptive Case Management (ACM'12)* Bd. 132. Springer, 2012, S. 31–42

[MR14] MUNDBROD, Nicolas ; REICHERT, Manfred: Process-Aware Task Management Support for Knowledge-Intensive Business Processes. In: IEEE COMPUTER

SOCIETY PRESS (Hrsg.): *IEEE 18th Int'l Distributed Object Computing Conference - Workshops and Demonstrations*. 2014, S. pp. 116–125

[MR17] MUNDBROD, Nicolas ; REICHERT, Manfred: Configurable and Executable Task Structures Supporting Knowledge-intensive Processes. (2017)

[Off16] OFFERGELD, Michael: *Skript der Vorlesung Usability Engineering*. Universität Ulm, 2016

[Sas18] SASS CONTRIBUTORS: *Sass Documentation*. `http://sass-lang.com/documentation/`. Version: 25.01.2018

[Sta18a] STATCOUNTER: *Browser Market Share*. `http://gs.statcounter.com/browser-market-share/desktop/worldwide`. Version: 02.02.2018

[Sta18b] STATCOUNTER: *Screen Resolution Stats*. `http://gs.statcounter.com/screen-resolution-stats/desktop/worldwide`. Version: 02.02.2018

[Vac11] VACULIN, R., HULL, R., HEATH, T., COCHRAN, C., NIGAM, A., SUKAVIRIYA, P.: Declarative business artifact centric modeling of decision and knowledge intensive business processes. In: *15th IEEE International Enterprise Distributed Object Computing Conference (EDOC), 2011*. IEEE, 2011, S. 151–160

[W3S18a] W3SCHOOLS: *Browser Display Statistics*. `https://www.w3schools.com/browsers/browsers_display.asp`. Version: 02.02.2018

[W3S18b] W3SCHOOLS: *Browser Statistics*. `https://www.w3schools.com/browsers/default.asp`. Version: 02.02.2018

[Wri17] WRIKE, Inc.: *Wrike*. `https://www.wrike.com`. Version: 12.11.2017

Name: Daniel Metzger Matrikelnummer: 756206

Erklärung

Ich erkläre, dass ich die Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Ulm, den .....

Daniel Metzger